THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *01/10/2020* **par :**

Renan Leroux-Beaudout

Méthodologie de conception de systèmes de simulations en entreprise étendue, basée sur l'ingénierie système dirigée par les modèles

---

JURY

Claude Baron	Professeur d'Université	Présidente du Jury
Antoine Beugnard	Professeur de l'IMT	Membre du Jury
Isabelle Borne	Professeur d'Université	Membre du Jury
Jean-Michel Bruel	Professeur d'Université	Membre du Jury
Julien Deantoni	Professeur d'Université	Membre du Jury
David Lugato	Ingénieur recherche (HDR)	Membre du Jury
Ileana Ober	Professeur d'Université	Membre du Jury
Marc Pantel	Maître de conférences	Membre du Jury

École doctorale et spécialité :
MITT : Informatique

Unité de Recherche :
Institut de Recherche en Informatique de Toulouse (UMR 5505)

Directeur(s) de Thèse :
Jean-Michel Bruel et *Ileana Ober*

Rapporteurs :
Antoine Beugnard et *Julien Deantoni*

Invité :
Guillaume Volbrecht - Chef de projet

Page intentionnellement blanche

À la mémoire
de ma compagne Marie-Hélène.

Remerciements

Je tiens à vivement remercier, mon responsable direct de l'époque, Monsieur Stéphane Rivet (Altran), pour avoir proposé ma candidature sur le projet MOISE-IRT Saint Exupéry. Occasion inespérée et effet du hasard, j'ai été le premier « mis à disposition » à faire cette demande de thèse à l'IRT. Il a fallu évaluer les impacts contractuels concernant l'IRT et les partenaires du projet MOISE. Il eût suffi d'un avis franchement négatif, pour que cette thèse ne puisse se faire. Encore aujourd'hui et pour moi, le montage de cette thèse et son acceptation est exceptionnelle.

Je tiens à présenter ma profonde gratitude à mon directeur de thèse Jean-Michel Bruel, qui a accepté le principe de cette thèse, puis accompagné dans ce projet, à Paul Chardon-Perrez, à l'époque, responsable du projet MOISE, qui s'est enthousiasmé à cette idée, et à Guillaume Volbrecht qui a repris cette initiative. Profonde gratitude, également, à ma co-directrice de thèse Ileana Ober, à Marc Pantel, pour leurs encouragements, leurs conseils avisés et pour leurs connaissances humaines et techniques des grandes entreprises industrielles locales.

Cette thèse n'aurait pu se faire sans l'approbation des directions et du département « Systèmes embarqués » de l'IRT Saint Exupéry, le tout représenté, à l'époque par son Président Monsieur Gilbert Casamatta et par Monsieur le Directeur général Ariel Sirat. Il en est de même pour l'accord des partenaires du projet MOISE qui ont encouragé de manière explicite ou implicite la réalisation de ce projet de thèse. Soyez assuré de ma profonde reconnaissance.

Merci aux membres du groupe cosimulation pour vos apports. Ils enrichissent, maintenant, cette méthodologie.

Lors des moments difficiles, lorsque mes ressources intérieures m'ont semblé épuisées, j'ai pu en découvrir ou en redécouvrir d'autres, oubliées, cachées, insoupçonnées, à travers l'observation des qualités humaines, de vous, mes collègues de l'équipe MOISE. De tout mon cœur, merci à chacune et à chacun.

Chaleureux remerciements à ma soeur Sabine, à Gilles, Virginie, Claudia, à mon cousin Jean-Paul, à sa femme Josette, à mes amis, qui m'avaient encouragé dans ce projet et soutenu durant sa réalisation. Votre présence m'est toujours aussi précieuse.

Aux personnes inconnues que j'ai croisé au cours de mes pérégrinations, à celles que je revois encore et que je ne reverrais probablement plus du fait des circonstances, à vous toutes avec lesquelles j'ai pu dialoguer, sourire, rire, vous m'avez, sans le savoir, apporter votre richesse humaine. Merci pour votre gentillesse.

Table des matières

Table des figures

1. *Model Driven Architecture*
2. *Ingénierie Système*

3. *Functional Mockup Interface*

Table des sigles et acronymes

AIDA	*Aircraft Inspection by Drone Assistant*
ALSP	*Aggregate Level Simulation Protocol*
API	*Application Programming Interface*
ARCADIA	*ARChitecture Analysis and Design Integrated Approach*
CESAM	*CESAMES System Architecting Method*
CORBA	*Common Object Request Broker Architecture*
CPS	*Cyber-Physical System*
DARPA	*Defense Advanced Research Projects Agency*
DCOM	*Distributed Component Object Model*
DGA	*Direction Générale de l'Armement*
DIS	*Distributed Interactive Simulation*
DoD	*Department of Defense (US)*
DSL	*Domain Specific Language*
DSML	*Domain Specific Modeling Language*
Ecore	*Eclipse Core*
EE	*Enterprise Étendue*
EFFBD	*Enhanced Function Flow Block Diagram*
EIA	*Electronic Industries Alliance*
EMF	*Eclipse Modeling Framework*
FMI	*Functional Mockup Interface*
FMU	*Functional Mockup Unit*
FOM	*Federation Object Model*

FTP *File Transfert Protocol*

FTPS *File Transfert Protocol Secure*

GEMOC *The GEMOC Initiative On the Globalization of Modeling Languages*

HLA *High Level Architecture*

HTTPS *HyperText Transfer Protocol Secure*

IA *Intelligence Artificielle*

IDM *Ingénierie Dirigée par les Modèles*

IEEE *Institute of Electrical and Electronics Engineers*

IETF *Internet Engineering Task Force*

IHM *Interface Homme-Machine*

INCOSE *International Council on Systems Engineering*

IP *Intellectual Property*

IPC *Inter-Process Communication*

IPNC *Inter-Process Network Communication*

IPM *Infrastructure Project Manager*

IS *Ingénierie Système*

ISO *International Organization for Standardization*

MARTE *Modeling and Analysis of Real-time and Embedded systems*

MBSA *Model-Based Safety Analysis*

MBSE *Model-Based Systems Engineering*

MDA *Model Driven Architecture*

MIC *Model Identity Card*

MOF *Meta-Object Facility*

MoI *Model of Intention*

MOISE *MOdels and Information Sharing in Extended Enterprise*

MVC *Model View Controller*

NCOSE *National Council on Systems Engineering*

OEM *Original Equipment Manufacturer*

OMG *Object Management Group*

ONERA *Office National d'Études et de Recherches Aérospatiales*

OS *Operating System*

OTAN *Organisation du Traité de l'Atlantique Nord*

PDM *Platform Description Model*

PID *Proportional Integral Derivative*

PIM *Platform Independent Model*

POC *Proof Of Concepts*

RFLP *Requirement, Functional, Logical, Physical view*

RFP *Request For Proposals*

RTI *RunTime Infrastructure*

S2C *System & Safety Continuity*

SDF *Synchronous Data Flow*

SEM *Simulation Execution Manager*

SiA *Simulation Architect*

SIMNET *Simulation Networking*

SMD *Simulation Model Developper*

SOM *Simulation Object Model*

SoS *System of System*

SRM *Simulation Reference Model*

SyA *System Architect*

SSL *Secure Sockets Layer*

SysML *Systems Modeling Language*

TFSM *Timed Finite State Machine*

TLS *Transport Layer Security*

TRL *Technology Readiness Level*

UML *Unified Modeling Language*

VPN *Virtual Private Network*

WAN *Wide Aera Network*

XMI *XML Metadata Interchange*

Chapitre 1

Introduction

Sommaire

1.1 Introduction

Ces travaux de thèse sur « *une méthodologie de conception de systèmes de simulations en entreprise étendue, basée sur l'ingénierie système dirigée par les modèles* » sont le produit d'évolutions successives, tant d'un point de vue technique, que sociétale. Cette introduction présente, dans les sections suivantes, les points clefs sur lesquels se fondent les propositions de cette thèse. La section 1.2 aborde les difficultés pour concevoir, vérifier et valider des systèmes complexes. Les difficultés de la section 1.2 peuvent être résolues par la construction de simulateurs. Or cette approche est source de questionnements quant à leur conception et à l'établissement des liens entre ceux-ci et les systèmes complexes à vérifier/valider (section 1.3). La notion de simulateur implique la présence de plateformes matérielles et logicielles de simulation (section 1.4). Cette section tire parti des concepts de l'ingénierie système et des modèles pour définir des éléments clefs à prendre en compte pour réaliser les plateformes de simulation en entreprise étendue.

Cette thèse est réalisée au sein de l'Institut de Recherche Technologique Antoine de Saint Exupéry et en collaboration avec l'Institut de Recherche en Informatique de Toulouse (IRIT). La section 1.5 décrit les raisons de l'existence de cet IRT, ses missions, ainsi que les principaux objectifs du projet industriello-académique MOISE, dans lesquels cette thèse s'inscrit.

1.2 Systèmes complexes

Un système est un ensemble d'éléments en interaction. Il est dit *compliqué* [res12] lorsque le nombre des éléments qui le constitue est significatif, et que l'analyse locale de chacun des constituants et de leurs interrelations permet de prédire son comportement global. À l'inverse, un système est considéré comme *complexe* [Gue19] [Hen+04] lorsque la prédiction de son comportement ne peut se faire par l'analyse locale de ses constituants et de leurs interrelations, mais uniquement par l'expérience ou la simulation de sa globalité.

L'ingénieur système en charge de la conception d'un système, qui peut être *compliqué* voire *complexe*, peut (se doit de) s'appuyer sur des méthodes d'ingénieries système en modélisant les exigences, l'architecture fonctionnelle et physique, s'il veut que le produit à réaliser soit conforme aux exigences du client. Cette modélisation lui permet de questionner ses architectures, d'en extraire des éléments d'analyse (par exemple le coût, la masse, etc.), de vérifier la cohérence entre les différentes vues métiers du produit (sûreté, sécurité, électrique, mécanique, hydraulique, etc.). Outre la prise en compte des contraintes métiers lors de la conception de l'architecture du produit, l'architecte système est encouragé, dans le cas de système *compliqué*, voire obligé dans le cas de système *complexe*, à réaliser des modèles dédiés à la simulation pour s'assurer du bon fonctionnement du produit, ou à tout le moins, d'une partie de celui-ci, en faisant intervenir les spécialistes des différents corps de métiers impliqués dans la réalisation de ce même produit. Ces spécialistes pouvant appartenir à différentes entreprises, et pour des raisons de confidentialité de la propriété intellectuelle, ces partenaires souhaitent, quelquefois, créer leurs propres modèles de simulation et réaliser la simulation au sein de leur entreprise respective. L'objet de cette thèse est également de développer une méthode pour structurer, organiser ces simulations inter-entreprises, dénommées *cosimulations* en entreprise étendue.

1.3 Difficultés associées aux modèles exécutables de simulation

Souvent, lors de la conception d'un produit, deux équipes de travail sont constituées, chacune dans leur domaine métier spécifique : architecture et simulation. Toutes deux se basent sur les exigences du produit à réaliser. La première centre son intérêt sur la conception de l'architecture du produit. La seconde vise à créer des modèles exécutables de simulation, pour vérifier et valider la faisabilité en regard des exigences du client. Cette approche permet, dans le cas d'un dialogue important entre les deux équipes, de mettre en exergue des exigences d'interprétation difficiles et de confronter les points de vues. Toutefois, cette manière de travailler pose le problème de la cohérence, de la fidélité, entre la définition de l'architecture du produit et de l'architecture de simulation. Ces deux architectures peuvent être différentes suivant l'interprétation des exigences faite par les deux équipes. Cette thèse propose une démarche différente : partir des exigences pour concevoir l'architecture du produit, puis, en s'appuyant sur cette architecture, créer une architecture de simulation qui intègre les corps de métier, c'est-à-dire les différents partenaires du projet avec leurs moyens propres de simulation, constituant de facto un ensemble de *cosimulation* en entreprise étendue. De cette manière, les écarts d'interprétations quant à l'architecture du produit sont supprimés, ou à tout le moins minimisés. Cette approche offre des avantages multiples. Il devient possible d'assurer une traçabilité, de bout en bout, entre l'architecture du produit, l'architecture de simulation, jusqu'aux modèles exécutables de simulation disponibles chez chaque partenaire impliqué dans le projet. Un autre atout important, de cette manière de faire, est de pouvoir créer des modèles exécutables de simulation en fonction des avancés dans la définition de l'architecture système du produit. Cela se traduit concrètement par la réutilisation possible des modèles exécutables de simulation déjà développés lors d'une simulation précédente, pour une exploration en *extension* comme en *profondeur* de l'architecture du produit. L'exploration en *extension* consiste en l'exécution de modèles de fonctions exécutables reliées entre elles, sur un même niveau d'abstraction. L'exploration en *profondeur* correspond en la simulation de fonctions *raffinées*. L'architecte système du produit peut, ainsi et à tout moment, explorer son architecture suivant une démarche matricielle, c'est-à-dire suivant une combinaison alliant l'*extension* comme la *profondeur*. Si l'approche proposée présente des avantages, elle est également sujette à un inconvénient. L'équipe en charge de la définition de l'architecture du produit peut mésinterpréter des exigences, sans avoir le garde-fou de l'équipe de simulation quant à l'analyse de ces mêmes exigences. Cet effet de bord est toutefois amoindri par l'objectif même de la simulation, qui est de vérifier/valider, au plus tôt, la faisabilité les exigences du client.

1.4 Ingénierie et modèles pour les simulateurs en entreprise étendue

Concevoir une *cosimulation* en entreprise étendue revient à définir un simulateur comme étant constitué par plusieurs composants : une architecture physique répartie sur différentes entreprises, une architecture logicielle dédiée et les modèles fonctionnels exécutables du produit à simuler. Le simulateur est conçu en fonction d'un objectif précis et spécifié par l'architecte système en charge de la définition du produit.

Concevoir un simulateur, plus exactement, son architecture matérielle et logicielle est un projet en soi, particulièrement si celui-ci s'inscrit dans le cadre d'une entreprise étendue, c'est-à-dire faisant intervenir d'autres partenaires industriels. À l'instar du développement d'un produit, cette thèse propose d'utiliser l'ingénierie système dirigée par les modèles pour concevoir la plateforme de *cosimulation*, en proposant un ensemble d'exigences issues de retours d'expériences, une analyse fonctionnelle et physique de la plateforme de *cosimulation*, avec les modèles associés. La partie matérielle concerne les unités d'exécution, les moyens de communication, les moyens de protection de sûreté et de sécurité. La partie logicielle, quant à elle, est centrée sur un ensemble de services nécessaires à la gestion de la documentation des modèles de simulation, à l'exécution de ces mêmes modèles de simulation, à l'exploitation des résultats et à la sécurité des accès. Cette architecture matérielle comme logicielle est conçue pour assurer la protection intellectuelle des différents partenaires de cette entreprise étendue. Afin de pouvoir répliquer cette plateforme de *cosimulation*, tout en tenant compte de la diversité des configurations, nous proposons deux métamodèles, l'un pour l'architecture matérielle, l'autre pour l'architecture logicielle.

1.5 Contexte de réalisation de cette thèse

1.5.1 Introduction

Cette section présente le contexte industriel de cette thèse, réalisée au sein de l'Institut de Recherche Technologique Antoine de Saint Exupéry, à Toulouse (cf. sous-section 1.5.2). Il héberge le projet MOISE, issu de la volonté de différents partenaires industriels et académiques de développer des méthodes et outils pour le partage des modèles en entreprise étendue. Ce projet est décrit dans la sous-section 1.5.3.

1.5.2 L'Institut de Recherche Technologique Saint-Exupéry

En 2010, le gouvernement français a initié un *plan d'investissement d'avenir* (PIA), doté de 35 milliards d'euros. Ce programme a été suivi par un deuxième plan de 12 milliards (2013), étendu, en 2017, par un troisième plan de 10 milliards d'euros. Ce programme a pour vocation « de stimuler l'emploi, de renforcer la productivité et d'accroître la compétitivité des entreprises françaises, en favorisant l'investissement et l'innovation dans des secteurs prioritaires, générateurs de croissance » [ANR10]. C'est dans ce cadre général que sont créés les IRT (Institut de Recherche Technologique), les IHU (Institut Hospitalo-Universitaire), les SATT (Sociétés d'Accélération du Transfert de Technologies), et que l'ANR (Agence Nationale de la Recherche) devient l'opérateur de l'état, gestionnaire des fonds, pour ce PIA (Plan d'Investissement d'Avenir), concernant l'enseignement supérieur et la recherche.

C'est en 2013 que l'IRT Antoine de Saint Exupéry fut fondé à Toulouse. Conformément aux objectifs du PIA, il a pour mission de développer des technologies compétitives et innovantes, dans les secteurs de l'aéronautique, du spatial et des systèmes embarqués, en interrelation étroite avec les industriels et la recherche académique. Ses domaines stratégiques de recherche sont les matériaux multifonctionnels de hautes performances, l'avion plus-électrique, les systèmes de communication intelligents et les systèmes embarqués. L'ensemble de ces travaux s'appuient sur des plateformes d'expérimentations technologiques [IRT19b].

Le domaine des matériaux multifonctionnels de hautes performances concerne les recherches sur les méthodes et les outils en lien avec les composites à matrice organique, céramique, les nanomatériaux, les traitements de surface. Le domaine de l'avion plus-électrique cherche à concevoir et introduire des équipements électriques plus légers, moins chers et plus fiables. Ces recherches s'appliquent également aux secteurs du spatial et de l'automobile. En juxtaposition du domaine précédent, et pour les mêmes secteurs susnommés, les systèmes de communication intelligents cherchent à répondre à l'introduction de l'IA (Intelligence Artificielle), et au besoin de communication par satellite pour améliorer les transferts de données. Enfin, le domaine systèmes embarqués promeut le développement de technologies, de méthodes et outils pour le traitement du signal, les systèmes autonomes/intelligents et l'ingénierie système collaborative. C'est dans ce dernier secteur que s'inscrit cette thèse, au sein du projet MOISE, dont un aperçu est donné dans la section suivante.

Toutefois, avant de passer à la description du projet MOISE, et pour donner un ordre d'idée du lien entre les académiques et l'industrie, l'IRT Saint Exupéry positionne ses travaux sur l'échelle des niveaux de maturité technologique TRL [1] [DoD10], entre les niveaux 3 (Fonction critique analytique et expérimentale et/ou preuve de concept caractéristique) et 6

1. *Technology Readiness Level*

(Modèle de Système/Sous-système ou démonstration de prototype dans un environnement pertinent). Ce positionnement permet de prendre en compte les résultats de recherches initiales des académiques (TRL de 1 à 3) et de fournir aux industriels des prototypes pertinents, sur lesquels ils pourront se baser pour aller vers une industrialisation possible (TRL de 6 à 9).

1.5.3 Le projet MOISE

Le projet MOISE (MOdels and Information sharing for System engineering in Extended enterprise) débute en janvier 2015 et se termine en mars 2019. Ce projet s'inscrit dans le département « Systèmes embarqués » de l'IRT Saint Exupéry, plus spécifiquement dans la section « Ingénierie système ». Ce projet a proposé et validé une approche de développement collaboratif, basée sur les modèles, dans un contexte d'entreprise étendue. Il est constitué de quatre groupes de travail : a) aider les interlocuteurs, dans une relation client-fournisseur, à exprimer leurs besoins à l'aide de l'ingénierie système basée modèles, b) assurer la cohérence entre la définition système et l'analyse de la sûreté/sécurité de ce même système, c) définir une architecture système pour la cosimulation en entreprise étendue, d) étudier une continuité numérique pour l'ingénierie système basée modèles.

Chaque groupe de travail est caractérisé par des enjeux industriels propres et des résultats attendus. Les enjeux du premier groupe sont d'améliorer l'adéquation de la solution système par rapport aux besoins du client, d'améliorer la qualité des interactions entre le fournisseur du système et le donneur d'ordre et d'enrichir la compatibilité entre les spécifications du client et la proposition technique provenant du fournisseur du système [BDF18a] [BDF18b]. Sur cette base, il en ressort des vues opérationnelles pour modéliser le contexte système, un processus d'analyse des besoins des parties prenantes (y compris les besoins du client), et un processus d'amélioration des spécifications du client. Les vues et le processus sont modélisés suivant la méthode CESAM [2] [CES19], au sein du logiciel CAPELLA [CAP19].

Quant au deuxième groupe, ses enjeux sont plutôt centrés sur la recherche d'une méthode pour assurer la cohérence entre l'analyse de la sûreté/sécurité et la conception du système [Alb+17] [Alb+18] [Pro+17], sur la possibilité d'obtenir des itérations efficaces pour la co-conception des systèmes, entre l'analyse de sûreté/sécurité et l'analyse système, sur le développement de processus compatibles des standards ARP4754A [SAE10] et ARP4761 [SAE96]. Les résultats obtenus portent sur la définition d'un processus de synchronisation entre les modèles d'analyse système et d'analyse de la sûreté/sécurité, la définition de point de vues, et enfin sur la démonstration de la faisabilité sur des développements industriels.

2. *CESAMES System Architecting Method*

Pour le troisième groupe, les enjeux s'expriment par la recherche d'une cohérence entre les architectures système et les activités de simulation, la prise en compte de l'hétérogénéité des modèles systèmes et des outils, l'évaluation des performances des analyses par la simulation, et de garantir la protection de la propriété intellectuelle à travers l'infrastructure réseau distribuée [Bos+18] [Ler+17] [Ler+18a] [Ler+18b]. De ces travaux auxquels cette thèse a contribué, il émerge un métamodèle pour la synchronisation entre les architectures système et de simulation, des exigences qualités pour la simulation des modèles, une méthodologie pour construire une architecture de référence pour la simulation à partir des modèles d'ingénieries système, et une preuve de concepts basée sur des DSL [3] proposés et mis en œuvre au sein du logiciel CAPELLA.

Enfin, le quatrième groupe se focalise sur la manière d'accélérer et d'étendre les analyses d'ingénierie des systèmes en entreprise étendue, en prenant en compte, les différents domaines métiers, l'hétérogénéité des méthodes et outils, la protection de la propriété intellectuelle et des échanges de données. Il en résulte la mise en place de la définition et évaluation des concepts clés permettant une ingénierie collaborative et une continuité numérique tout au long de la vie du produit dans l'entreprise étendue. Une preuve de concept (POC [4]) a été créée, sous la forme d'une plateforme collaborative, démontrant qu'il est possible d'avoir une continuité numérique dans l'entreprise étendue.

1.6 Vue d'ensemble des chapitres du manuscrit

Cette introduction générale a présenté les difficultés de haut niveau à résoudre et les solutions proposées pour répondre aux préoccupations des ingénieurs système en charge de la réalisation d'un produit.

Cette thèse se veut inscrite dans une sorte de continuité historique. Le chapitre sur l'état de l'art (chapitre 2) rappelle que les Hommes ont, très tôt, modélisé certains aspects du monde dans lequel Ils vivent, pour tenter d'en comprendre son fonctionnement et d'en prévoir ses manifestations (section 2.4). De par sa nature limitée, il a également cherché à réduire ses efforts en inventant des solutions technologiques, pour se déplacer, pour se nourrir, pour construire, ou encore pour produire les biens qui lui sont importants pour vivre (ou qui peuvent lui sembler importants) (sections 2.2 et 2.3). L'ensemble de ces activités nécessite une collaboration entre sachants de divers métiers. Elle peut se traduire, dans nos sociétés modernes, par la notion d'entreprise étendue (section 2.5). Enfin et outre la présentation d'un panorama de méthodes et d'outils dédiés à l'ingénierie système (sections 2.6 et 2.7.7), cet état de l'art décrit l'apport fécond des concepts de l'ingénierie

3. *Domain Specific Language*
4. *Proof Of Concepts*

dirigée par les modèles à l'ingénierie système (section 2.7), dans le but de vérifier et valider la conception du produit par la simulation (sections 2.8 et 2.9).

Les contributions/propositions de cette thèse (chapitre 3) s'articulent suivant deux axes. Le premier concerne la méthodologie (partie gauche de la figure 1.1) à employer pour créer les modèles de simulation (section 3.2). Elle part de la définition de l'architecture système du produit et se développe jusqu'à l'exécution des modèles de simulation. Cette méthodologie intègre trois nouveaux acteurs, dont le premier : l'architecte de la simulation se positionne entre l'architecte système pour la définition du produit et les développeurs des modèles de simulation. Le deuxième nouvel acteur prend en charge les modèles de simulation, en provenance des développeurs, pour en assurer l'exécution au sein de chaque entreprise partenaire. Le deuxième axe (partie droite de la figure 1.1) s'attache à définir, suivant une démarche d'ingénierie système dirigée par les modèles, ce que peut être une plateforme de cosimulation en entreprise étendue. Cette approche intègre un troisième nouvel acteur en charge de la réalisation de la plateforme de simulation au sein de son entreprise : l'IPM (Infrastructure Projet Manager). À ce nouvel acteur s'ajoute un ensemble d'exigences centrées sur l'architecture de la plateforme, sur les moyens de communication, sur les logiciels et matériels nécessaires à la mise en œuvre opérationnelle de la plateforme de cosimulation (section 3.3). Le tout est accompagné d'une analyse fonctionnelle et physique, et en dernier ressort, par une proposition de deux types d'abstractions pour réaliser cette plateforme de cosimulation. Après l'avoir instanciée (section 3.4), cette plateforme doit être vérifiée, puis validée, avant de pouvoir exécuter les modèles de simulation (section 3.5). Cette thèse étant réalisée au sein de l'équipe MOISE, ces propositions se sont nourries des retours d'expérience de cette même équipe et inversement (section 3.6). Enfin, sans entrer directement dans le travail de thèse et pour conserver les éléments de réflexion sur les clauses de confidentialité, la gestion de la propriété intellectuelle pour les entreprises entrantes, comme sortante du partenariat, sur les exigences centrées sur les servitudes, une section est dédiée aux contributions à destination des industriels (section 3.7). Notre méthodologie pour créer des simulateurs est validée (chapitre 4) via un cas d'utilisation, développé au sein de l'équipe MOISE : le drone AIDA (section 4.2), tout en étant remanié pour les besoins de cette validation. Ce cas d'étude a la particularité d'être suffisamment dense pour éviter l'écueil de la simplicité. Pour montrer la souplesse de cette méthode, notre validation s'appuie sur la mise en œuvre de la méthodologie CESAM (sous-section 2.7.5) par l'architecte système en charge de la définition du produit et par la méthode RFLP [5] (sous-section 2.7.4) par l'architecte de la simulation, en suivant l'approche matricielle définie en sous-section 3.2.3. La validation se fait à la fois en *extension* comme en *profondeur*. Cette validation a permis de mettre en exergue, qu'il est possible de réutiliser les modèles définis lors de phases d'analyses précédentes *amont* ou *aval*, et que l'application de cette méthode tend bien à rationaliser le développement des modèles de simulation.

5. *Requirement, Functional, Logical, Physical view*

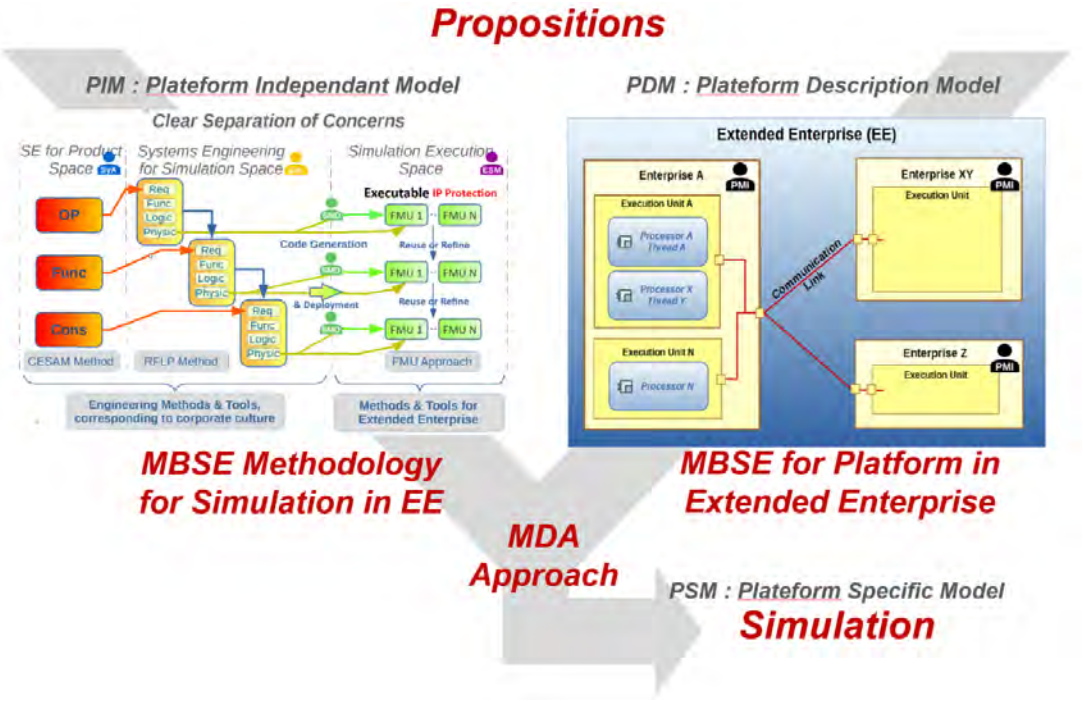

FIGURE 1.1 – Vue d'ensemble des propositions, avec l'approche MDA.

Ces chapitres sont suivis par une conclusion et des perceptives (chapitre 5) quant à l'emploi de cette méthodologie à destination des *Systèmes de systèmes*, ou encore à la partie basse du cycle en V.

Chapitre 2

Etat de l'art

Sommaire

2.1 Introduction

Ce chapitre a vocation à dresser un état des lieux de la littérature scientifique, en rapport avec les questions soulevées en introduction. Il aborde certaines évolutions qui ont conduit aux types de productions de l'on connaît aujourd'hui. Parmi celles-ci, la première section fait référence à la production artisanale jusqu'à la production de masse de l'usine 4.0. La section suivante décrit les différents types de sous-traitances qui sont apparus pour résoudre les difficultés liées à cette production de masse. La troisième section précise les concepts de *compliqué* et *complexe*. Ces deux notions se sont révélées constantes au cours des siècles, même si les objets, sur lesquels elles portent, ont varié en fonction des découvertes scientifiques.

La section sur le concept d'entreprise étendue (section 2.5) décrit les enveloppes juridique et organisationnelle de deux types de regroupement d'entreprises pour réaliser un projet commun. Le premier est de type « Donneur d'ordres à sous-traitants », le deuxième, à mettre en regard du premier, est de type « Réseau nucléique ».

Ce chapitre sur l'état de l'art restitue dans le contexte de l'époque, l'émergence de

l'ingénierie système et précise le contour et les éléments importants du standard d'IS EIA [1]-632 (section 2.6).

L'ingénierie dirigée par les modèles (IDM) est abordée (section 2.7) pour déterminer et caractériser son influence sur l'ingénierie système dirigée par les modèles (ISDM), dont différentes méthodologies ISDM sont décrites dans cette même section 2.7. La section sur la cosimulation explore différents types de simulation, les problèmes dans la gestion du temps et deux standards de communication (HLA [2] et FMI) pour réaliser des simulations réparties entre partenaires. La section 2.9 porte sur une évolution de la pensée quant à la prise en compte des modèles de simulation hiérarchique et hybride. Cette évolution apparaît dans l'étude des concepts mis en œuvre dans les logiciels de simulation (dans l'ordre) Modelica, Ptolemy, ModHelX et Gemoc Studio.

Pour commencer cet état de l'art, la section suivante 2.2 aborde de manière très synthétique une vision de monde de la production, sous l'angle de « *Savoir d'où l'on vient* ». Cette section tente de mettre en avant *l'Homme*, avec ses préoccupations associées à la notion de production et son évolution.

2.2 Évolution de la production

La figure 2.1 tente de traduire, avec un regard synthétique, les évolutions des moyens de production et de la complexité des produits, avec cette idée constante et sous-jacente de réduire les coûts de production, quelles que soient les périodes. Suivant les époques, les coûts de production peuvent être vu comme les dépenses énergétiques musculaires des Hommes, ou encore monétaire, l'un n'étant pas exclusif de l'autre, bien évidemment.

Par exemple, au moyen âge et pour la construction de la cathédrale de Paris, l'idée novatrice a été de faire tailler les pierres de construction directement sur le site d'extraction. De cette manière, le coût du transport était réduit, puisque seule la pierre prête à l'emploi était transportée. Concevoir un tel édifice, un tel système (du latin « *systema* ») posait déjà des problèmes d'organisation, de construction, en regard des outils et technologies disponibles. Ce genre de réalisation unique, en faible quantité, de grandes diversités, nécessite un personnel qualifié, de type artisanal [Fon99]. Cela s'applique aussi bien aux fameux meubles d'André-Charles Boulle (1642 - 1732), aux violons d'Antonio Stradivari (1644 - 1737), à l'horloge astronomique et aux automates de la cathédrale de Strasbourg (1352 - 1547 - 1838).

Bien plus tard, toujours dans la perspective de réduire les coûts d'extraction du charbon,

1. *Electronic Industries Alliance*
2. *High Level Architecture*

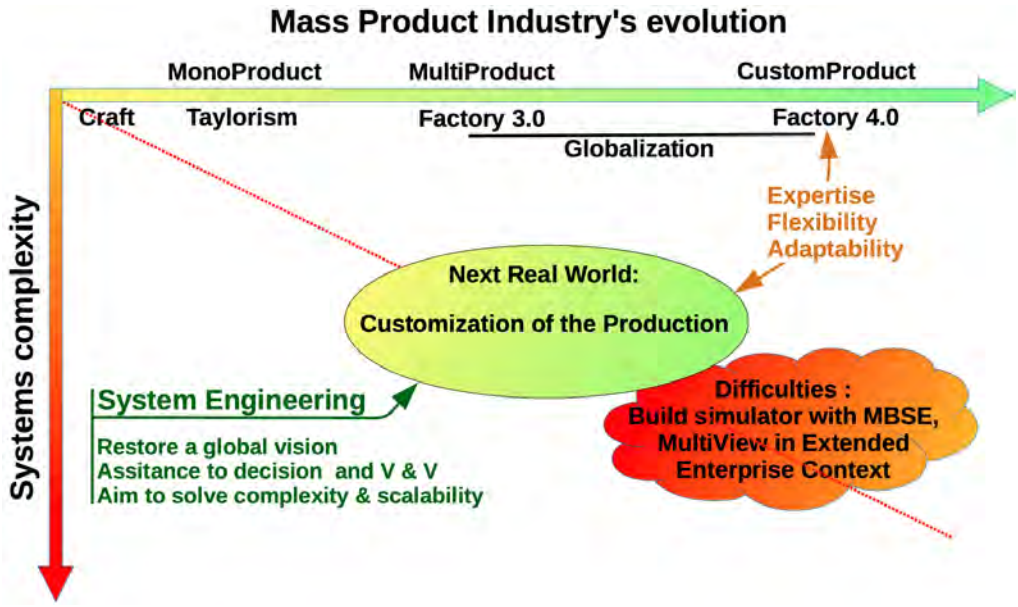

FIGURE 2.1 – Évolution de la production de masse

James Watt améliora significativement le moteur à vapeur de Thomas Newcomen. C'est le début de la première révolution industrielle (1784), créée en Angleterre, avec la maîtrise des énergies hydrauliques et fossiles et la disponibilité de ces nouveaux moteurs à vapeur.

C'est au cours de la deuxième révolution industrielle (électricité et convoyeur - 1870) que Taylor crée la division du travail suivant deux axes (1890). L'axe vertical prône la séparation entre la conception et la réalisation : aux ingénieurs - la conception, les gammes de fabrication ; aux ouvriers - les tâches d'exécution. L'axe horizontal cherche à répartir de manière optimale et sans doublon les tâches entre les différents postes de travail. Conséquences : augmentation de la productivité et sentiment de perte de compétences de la part des ouvriers (anciennement artisans), puisque les gestes associés à la production sont définis en amont [Bla12]. Quelques années plus tard, Henry Ford reprend les idées de Taylor et les applique au travail à la chaîne pour la production de la fameuse voiture « Ford T ». C'est la période de « *production de masse mono-produit* » durant laquelle, pour que l'entreprise existe, il suffit de « *Produire puis vendre* » [Fon99]. Cette période s'étend de la fin de la Première Guerre mondiale jusqu'à 1975, juste après le premier choc pétrolier. Elle est suivie de la période post-choc pétrolier : 1975-1985 « *Produire ce qui sera vendu* ». Elle se caractérise par la prévision commerciale, la gestion des approvisionnements et la régulation des stocks. Depuis 1985 à nos jours, c'est le « *Produire ce qui est déjà vendu* » [Fon99].

En parallèle et à partir des années 70, les usines ont intégré les automates, puis les robots, pour continuer à réduire les coûts de production et assurer une répétabilité des

résultats. Cette arrivée des robots peut s'apparenter à la troisième révolution industrielle [FIM15], avec laquelle des produits multiples et de même famille peuvent être réalisés sur une même ligne d'assemblage.

Cette réduction des coûts est devenue encore plus prégnante, singulièrement à partir des années 1980. À cette époque, il y a eu l'arrivée fulgurante de la mondialisation. Plus exactement, d'une mondialisation plurielle [Roc01] : *la mondialisation financière*, *la mondialisation de la division du travail.*

Aujourd'hui, la quatrième révolution industrielle commence à se mettre en place, en s'appuyant sur les technologies de la numérisation, de l'internet industriel, de la conception virtuelle. L'objectif affiché de cette révolution est de concevoir des usines de type agile, c'est-à-dire disposant de modes de production flexible, pour « servir » le client avec des produits individualisés. Pour ce faire et devant la complexité des produits, l'usine 4.0 intègre dès le départ les fournisseurs et clients, tout en se basant sur des maquettes virtuelles du produit et sur la simulation [FIM15]. Il devient de plus en plus difficile, à cette usine 4.0, de maîtriser l'ensemble des techniques nécessaires à l'outil de production, comme à la création de ces produits toujours plus technologiques. Pour pallier ces difficultés, le recours à une sous-traitance de spécialité plus importante semble s'imposer. Une évolution de celle-ci en est décrite dans la section suivante 2.3. La prise en compte des partenaires dans le processus de production est le prélude à une potentielle nouvelle forme d'entreprise : l'entreprise étendue, dont différentes structures sont décrites dans la section 2.5.

2.3 Évolution de la sous-traitance

Si la sous-traitance existait avant les années 80, c'est à partir de ces années-là que cette pratique s'est généralisée [Tin+07]. Cette sous-traitance prend plusieurs formes. La première, la sous-traitance de type « *capacité* » a pour vocation à répondre occasionnellement à une pointe de commandes, à un incident technique, ou encore, à préserver une capacité de production du donneur d'ordres, en faisant sous-traiter une partie de sa production [INS18a]. Lorsque le donneur d'ordre ne souhaite pas, ou ne possède pas les connaissances ni les machines pour répondre à son besoin spécifique, il a la possibilité de faire appel à une entreprise spécialisée dans le domaine désiré. Ce type de démarche est dénommée « *sous-traitance de spécialité* » [INS18b]. Une troisième forme de sous-traitance dite de « *substitution* » « *est motivée par le différentiel de prix entre le donneur d'ordre et le sous-traitant, qui ne résulte pas ici d'un différentiel de coût de production ou d'un défaut de compétences en tant que telles, mais d'un contournement des conditions d'emploi qui prévalent dans l'entreprise donneur d'ordres.* » [Tin+07]. Dans le principe, ce dernier type de sous-traitance ne semble pas être exclusif des deux premières formes.

Suite au choc pétrolier de l'année 1974, les entreprises ont cherché à se restructurer, à se recentrer sur leur cœur de métier, par l'externalisation d'une partie de leurs activités, pour réduire leurs coûts de production nécessaire à la compétitivité de l'entreprise et à la rentabilisation des capitaux qui y sont investis. J. Garnier ([Gar13]) définit l'externalisation comme consistant en « *l'abandon d'une activité ou d'une opération jusque-là effectuée au sein de la grande entreprise et, concomitamment, dans le recours à cette activité, sous une forme marchande contractualisée, auprès d'une entreprise spécialisée extérieure* ».

À la fin des années 80, le nombre de sous-traitants était devenu prohibitif. Leur coordination, par le donneur d'ordres, posait des difficultés, de par leur nombre, de par la diversité technique des éléments et des interactions entre ces mêmes éléments [Joe09]. C'est la raison pour laquelle, le donneur d'ordres a cherché à réduire ce nombre, en confiant des sous-ensembles de plus en plus complexes, pouvant être dénommés modules, à de la sous-traitance de spécialité [Joe09]. De facto, avec ces délégations, le métier du donneur d'ordre change. Son métier initial : d'architecte, de concepteur, de réalisateur et d'assembleur, évolue vers l'architecture et l'intégration (cf figure 2.2).

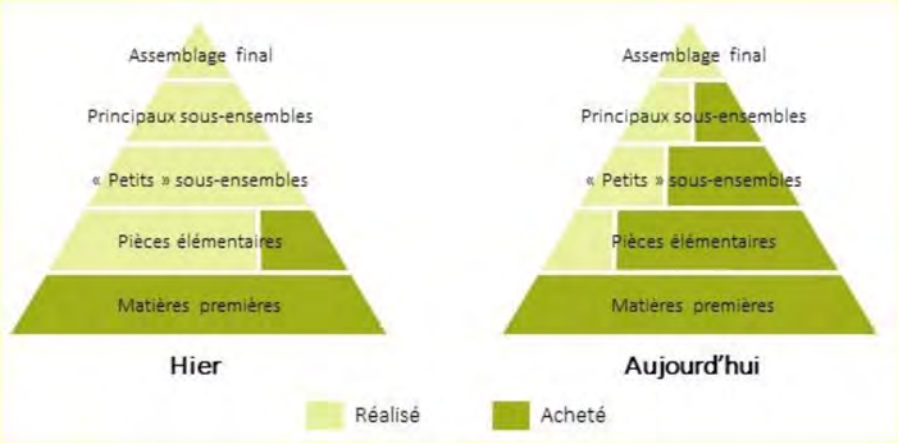

FIGURE 2.2 – Évolution du modèle organisationnel entre Airbus et ses sous-traitants (source Airbus 2012) [Exe15]

Du fait de cette externalisation, les connaissances technologiques nécessaires à la réalisation des modules se déplacent chez les sous-traitants. Le donneur d'ordres est amené à ne plus maîtriser cette complexité technique. Or, si ce même donneur d'ordre désire être toujours compétitif sur son marché, il se doit d'innover sur des technologies pouvant être intégrées dans les modules sous-traités. L'insertion de nouvelles technologies (ex : électronique de puissance, composites, motorisation des trains d'atterrissage) peut modifier de manière substantielle les interfaces entre les différents modules et altérer les interdépendances entre les différents systémiers [Joe09]. Cette connaissance lui est absolument nécessaire si le donneur d'ordres veut continuer à maîtriser les conditions de sa production, et conserver son attractivité commerciale.

L'une des manières de définir le périmètre des modules, de définir les interfaces et les interrelations entre ces modules est d'intégrer les partenaires, au sein d'un « plateau de co-développement », dès la phase de conception amont du produit [Joe09] [Exe15]. Cette approche permet au donneur d'ordres de s'assurer que l'architecture du produit est optimale, que les technologies concurrentielles sont bien intégrées dans les modules, et que les interactions entre les modules sont bien définies.

Dans une vision plus large que le « plateau de co-développement » qui laisse le sentiment d'une référence à un lieu particulier, il semble intéressant d'utiliser, en lieu et place, le terme d'« entreprise étendue ». Ce dernier terme suppose la mise en place d'une structure juridique et fonctionnelle, entre les différents partenaires, pour le temps de la réalisation d'un projet commun. Cette structure d'« entreprise étendue » a vocation à favoriser les échanges entre les partenaires, dans le respect d'une protection de la propriété intellectuelle librement définie et consentie, entre le donneur d'ordre et les partenaires, et entre les partenaires eux-mêmes. Or, [Cha12] décrit les relations des donneurs d'ordre vis-à-vis des sous-traitants comme : « le pillage du savoir-faire des PME est devenu un sport national ». Dans ce contexte, le problème de la protection des droits intellectuels est un sujet éminemment important pour qui cherche à protéger et faire valoir ses droits de sous-traitants. Elle l'est encore plus, avec l'arrivée de la financiarisation de la sous-traitance. C'est-à-dire, lorsque le donneur d'ordres demande aux sous-traitants systémiers de participer à la mise en œuvre d'un nouveau produit, d'un nouveau programme, avec tous les aléas commerciaux, dont le systémier n'est pas maître. Dans ce cas, la protection intellectuelle devient cruciale si celui-ci veut pouvoir espérer un retour sur investissement, en vendant au donneur d'ordres, comme aux concurrents du donneur d'ordres, des équipements issus de sa propre recherche. [MGC19] propose une approche de protection IP, basée modèles, bien adaptés aux différents partenaires qui œuvre au sein de l'entreprise étendue, lors de la phase de conception amont du produit. Cette approche peut restreindre la visualisation d'éléments de modèles, aux seules catégories des personnes autorisées, et surtout elle permet de marquer les éléments des modèles pour signifier le propriétaire. Ce point sur la propriété intellectuelle n'est pas plus développé dans cette thèse, autre que sous la forme de l'utilisation d'un intergiciel de type FMI 2.0. Ce standard FMI 2.0 spécifie, en autre, l'API[3] pour les échanges de données, sans nécessité de dévoiler le code source à l'origine de cet exécutable.

Cette notion d'« entreprise étendue » ne concerne pas uniquement une structure de type hiérarchique : donneur d'ordres à sous-traitant. Elle est bien plus riche, en ce sens où elle prend en compte des structures plus diverses, plus horizontales. Le concept d'entreprise étendue est abordé dans la section 2.5.

3. *Application Programming Interface*

2.4 Gestion de la complexité

Dans cette section, nous nous intéressons aux notions de *compliqué* et *complexe*. En reprenant des éléments de la section 2.2, la construction de la cathédrale de Paris a été une activité longue, difficile, et *compliquée*. Difficile dans le sens où cette construction a demandé beaucoup d'effort. Elle débute 1163 et se termine au milieu du XIVe siècle. Avec cet édifice, le *compliqué* a été de déterminer l'emplacement des arcs-boutants, de déterminer la répartition des forces et autres facteurs, pour assurer la stabilité de la nef. En d'autres termes, de déterminer les interactions entre les différents facteurs que sont les hauts murs, le poids de la nef et ce qu'il fallait de contre-poussé pour ériger les arcs-boutants aux emplacements les plus appropriés.

La notion de *complexité* est de toute autre nature. Elle est présente lorsque nous ne comprenons pas encore les interactions entre différents éléments. Par exemple, chercher à comprendre les lois de la nature reste encore aujourd'hui une activité complexe (astronomie, composition de la matière, psychologie, etc.). Dans le passé, la sphère armillaire est un bon exemple de tentative de modélisation pour comprendre le déplacement apparent complexe des planètes, qui a l'époque était placée dans une vision centrée et circulaire. Il a fallu attendre que Kepler publie ses deux lois sur les trajectoires elliptiques des planètes (1609), puis Newton avec la loi de la gravité universelle pour lever cette complexité, sur les déplacements des planètes et pleinement comprendre les interactions entres celles-ci. Plus tôt, à l'époque des Grecs, il y eut également des modélisations sur le déplacement complexe des planètes. L'une des plus employées fut les tables de Ptolémée. Elles servirent pour le calcul de la position des planètes, tant pour les astronomes que pour les navigateurs. Son utilisation ne fut définitivement abandonnée, par l'église, que vers 1750 [Wik19b].

Si jusqu'à présent, cette complexité concerne la «Nature », avec la deuxième révolution industrielle, les avancées technologiques, l'arrivée des automates, l'appel aux sous-traitances de spécialité, puis l'usine 4.0, il devient de plus en plus difficile de maîtriser, de comprendre l'ensemble des interactions et des effets secondaires des systèmes entre eux. L'Homme finit par produire sa propre complexité technologique qui s'ajoute à la complexité naturelle.

Pour ce qui est des activités humaines, la figure 2.1 tente de traduire cette l'évolution de la production de masse et de la complexité associée. Au niveau de la production de type artisanal (Craft), point de complexité, mais de la difficulté et du compliqué (Exemple : la cathédrale de Paris). L'arrivée du Taylorisme, c'est à dire de la production monoproduit, il est encore possible de considérer cette production de masse comme compliquée. En revanche, comment maîtriser l'ensemble des interactions entre les systèmes mécaniques, politiques, sociétaux, avec l'émergence des automates, de la sous-traitance de spécialité,

de l'IA [4], de la mondialisation, de l'uberisation, de l'usine 4.0 ? Cela semble bien être un problème complexe.

En se restreignant au niveau des systèmes physiques, pour espérer en comprendre les interactions, le besoin s'est fait de plus en plus prégnant de modéliser, de simuler, avec l'aide des acteurs impliqués, certaines de ces interactions, pour essayer d'en comprendre les mécanismes sous-jacents et maîtriser cette complexité. Cette compréhension par la modélisation et la simulation permet de structurer la conception et la construction de ces systèmes physiques. Cette approche par la modélisation et par la simulation a d'autant plus de sens, lorsque cela concerne de grands programmes ou ensembles industriels (figure 2.1), où les modèles sont en fait des *multi-modèles hétérogènes*.

2.5 Entreprise étendue

Pour [Ror05], schématiquement, il y a trois périodes stratégiques dans l'histoire des organisations des entreprises. L'organisation verticale, c'est-à-dire faire seule. Ce type d'organisation peut être illustré par des constructeurs d'automobiles, jusque dans les années 70. La deuxième période est caractérisée par le faire-faire, c'est-à-dire, faire appel à de la sous-traitance de capacité ou de spécialité. Enfin, l'entreprise réseau, pouvant également être dénommée « *entreprise étendue* » (EE [5]), peut être une réponse organisationnelle et juridique en regard de la globalisation, de la concurrence ou aux défis de nouveaux programmes industriels [Ror05]. Pour typer les entreprises étendues, [Ror05] définit 7 critères différenciants, qui sont :
1) la dynamique de création, c'est-à-dire la démarche qui a prévalu à la création de EE (réorganisation interne, association),
2) la logique constitutive qui se réfère aux principes de création de valeurs : additive (mise en commun d'activités similaires) ou complémentaire (ensemble d'activités complémentaires concourant à la création de valeurs),
3) l'agencement des enveloppes juridiques et organisationnelles dont les formes peuvent être inclusive, en juxtaposition ou en dissociation,
4) la stabilité qui caractérise la pérennité des partenaires (forte ou faible),
5) la structure organisationnelle de type projet ou fonctionnelle,
6) la distribution du pouvoir qui peut être polycentré, descendant, latéral ou ascendant,
7) les liens d'interdépendances entre les entreprises, par exemple de type communauté, c'est-à-dire dépendant d'une même ressource, de type séquentiel : une entreprise attend que l'entreprise précédente ait terminé son travail pour prendre la suite, et de type réciproque : l'action d'une entreprise déclenche l'action d'une autre qui, à son tour, réamorce la pre-

4. *Intelligence Artificielle*
5. *Enterprise Étendue*

mière.

De ces 7 critères, [Ror05] extrait 6 typologies d'entreprises étendues :
a) le réseau interne, fait suite à une démarche de réorganisation interne, qui regroupe des enveloppes organisationnelles au sein d'une même entité juridique sans modifier le périmètre extérieur de l'entreprise,
b) le réseau intégré correspond à une démarche d'externalisation, avec une seule enveloppe organisationnelle et plusieurs enveloppes juridiques,
c) le réseau pendulaire fait suite à une démarche d'externalisation des activités périphériques, et dont les personnels externalisés sont mis à disposition de l'entreprise initiatrice de cette externalisation.
d) le réseau fédéré correspond à un donneur d'ordre qui fédère autour de lui, un ensemble d'entreprises sous-traitantes, avec recomposition des enveloppes juridiques et organisationnelles des entreprises concernées.
e) le réseau nucléique se caractérise par la présence d'une entreprise noyau, autour de laquelle différents partenaires participent pour la durée d'un projet, chacun conservant son identité juridique et organisationnelle.
f) enfin, le réseau confédéré est constitué d'entreprises généralement concurrentes qui s'associent pour la durée d'un projet ou pour la pénétration d'un marché [Ror05].

Pour la suite de la thèse et pour être représentatif des relations entre les grandes entreprises locales et leurs partenaires, c'est un réseau de type fédéré qui est traité, c'est-à-dire du donneur d'ordres vers les sous-traitants. Ce réseau présente des caractéristiques qui lui sont propres. Sur la base des 7 critères énoncés par [Ror05], cela donne la physionomie suivante : la dynamique de création est caractérisée par l'externalisation. Ce qui signifie que chacun, des sous-traitants et le donneur d'ordres, possède sa propre enveloppe juridique. La logique constitutive est de type complémentaire. Ici, cette complémentarité correspond à une sous-traitance de spécialité (figure 2.3). L'agencement des enveloppes est dissocié. Le donneur d'ordre a scindé son enveloppe juridique de son enveloppe organisationnelle. Cette enveloppe organisationnelle englobe maintenant les sous-traitants (figure 2.3). La stabilité de ce réseau est considérée comme forte, dans la mesure où le réseau est bâti sur le long terme, pour réaliser le produit final. La structure organisationnelle est fonctionnelle, centrée autour de la réalisation du produit. Comme pour tous les donneurs d'ordres, la distribution du pouvoir est descendante et les liens d'interdépendances sont séquentiels.

La typologie réseau nucléique semble bien correspondre à l'IRT Saint Exupéry (figure 2.4). L'IRT est un centre de compétence dédié à la recherche technologique. La dynamique de création de ce genre de réseau est de type assemblage, dans le sens où l'IRT mobilise des partenaires pour la réalisation d'un projet précis [Ror05], ou encore sur leur demande. Un exemple en est donné par le projet MOISE, décrit dans le premier chapitre. La logique constitutive est additive, de par la mise en commun des compétences de recherche, de chaque partenaire, en vue d'obtenir un avantage concurrentiel pour chacun d'eux [Ror05].

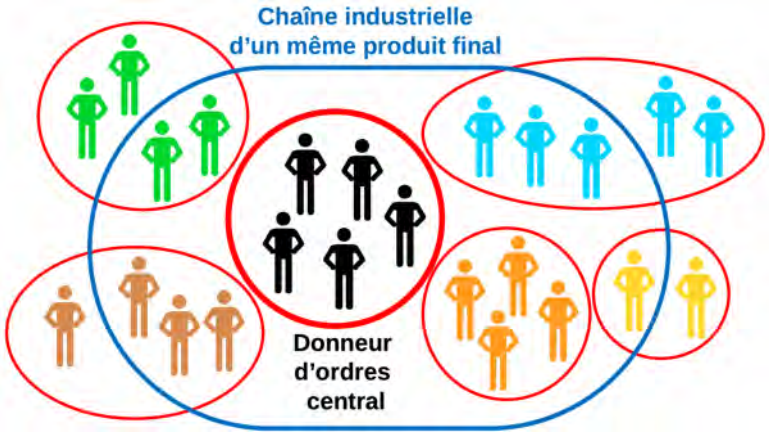

FIGURE 2.3 – Réseau fédéré [Ror05] (crédit : Boris ROBERT - équipe MOISE).

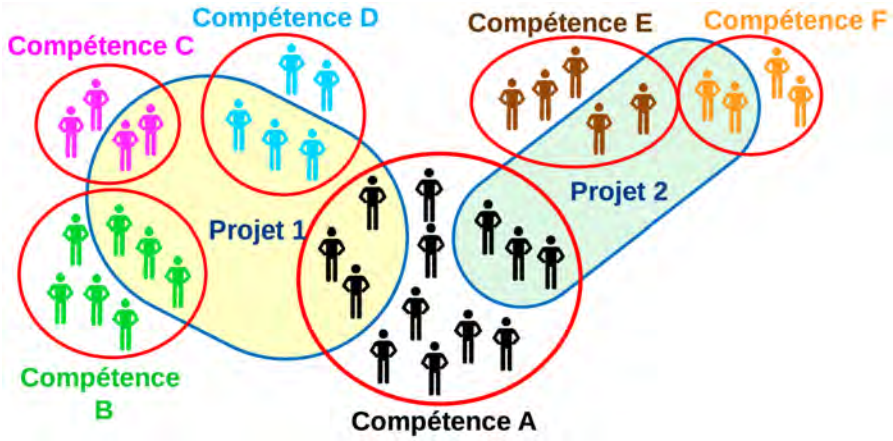

FIGURE 2.4 – Réseau nucléique [Ror05] (crédit : Boris ROBERT - équipe MOISE).

L'agencement des enveloppes est en juxtaposition. C'est-à-dire une juxtaposition de plusieurs enveloppes organisationnelles (les projets) au croisement de plusieurs enveloppes juridiques [Ror05] (l'aspect juridique de chaque partenaire). La stabilité est faible dans le sens où il est toujours possible à des partenaires d'entrer ou de sortir du projet. Comme illustré par la figure 2.4, la structure organisationnelle est de type projet. Dans cette organisation, la distribution du pouvoir est répartie horizontalement selon des rapports de force équilibrés entre les différents partenaires [Ror05]. Les liens d'interdépendance sont réciproques.

2.6 Ingénierie système

Pour tenter de gérer cette complexité technologique, l'ingénierie système apparaît vers 1937, pour mettre en place un système de défense aérienne [INC15]. À cette même époque, le principe de la pensée systémique apparaît au MIT [Don04]. Toutefois, « *c'est seulement dans les années 1970 que la nouvelle pensée prend véritablement son essor* » [Don04]. C'est à cette époque que naît la science des systèmes (ou systémique) dixit Le Moigne[Don04]. L'NCOSE [6] a été créé en 1990. Cet organisme s'ouvre à l'international et devient l'INCOSE [7] en 1995. L'INCOSE retient l'année 1969 comme date importante pour l'ingénierie système par la réalisation du premier standard Mil-Std 499. L'ingénierie système est formellement reconnue, en 2002, avec la création du standard international ISO/IEC 15288 en 2002 [INC15].

L'ingénierie système devient une démarche de moyens et une méthodologique interdisciplinaire, pour réaliser des systèmes performants avec succès [INC15]. Elle vise à résoudre les problèmes compliqués, le passage à l'échelle et à redonner une vision globale du produit aux parties prenantes. C'est aussi une aide à la décision, à la vérification et à la validation. Elle s'applique dès l'expression du besoin des utilisateurs, à la conception, aux interactions avec les autres systèmes, à la mise en service, à la maintenance du système et jusqu'à son retrait de service. Elle prend en compte l'ensemble de l'écosystème nécessaire au cycle de vie : la structure de l'entreprise qui réalise le système, la création et management du projet, les approvisionnements, les relations avec les personnes intervenant sur le système, les sous-traitants, les contraintes normatives et environnementales, les marchés. On retrouve bien dans cette démarche, l'approche de la pensée systémique dont elle est issue. La figure 2.5 propose une vue d'ensemble des standards qui ont été créés pour répondre à ces demandes, de gestion des systèmes compliqués et complexes.

Le premier standard militaire MIL STD 499, est créé en 1969 (année du premier pas de l'Homme sur la Lune). En partant de ce standard d'ingénierie système, création de deux normes civiles EIA-632 [EIA99] et IEEE [8] 1220 [IEE05], suivi par la norme ISO [9] 15288 en 2002 [IEE15]. Ces trois normes ne possèdent pas les mêmes périmètres d'intérêt (figure 2.6). La norme IEEE 1220 centre l'intérêt d'utiliser l'IS sur la spécification des besoins jusqu'à la définition physique du système. La norme EIA-632 se veut plus large, notamment dans la prise en compte de la partie intégration unitaire, équipements, vérification/validation, jusqu'au transfert vers l'utilisateur final, pour exploitation. La norme ISO 15288 prend en compte l'existence des normes précédentes et comble le reste du cycle de vie du système, c'est-à-dire, la partie exploitation et le retrait de service.

6. *National Council on Systems Engineering*
7. *International Council on Systems Engineering*
8. *Institute of Electrical and Electronics Engineers*
9. *International Organization for Standardization*

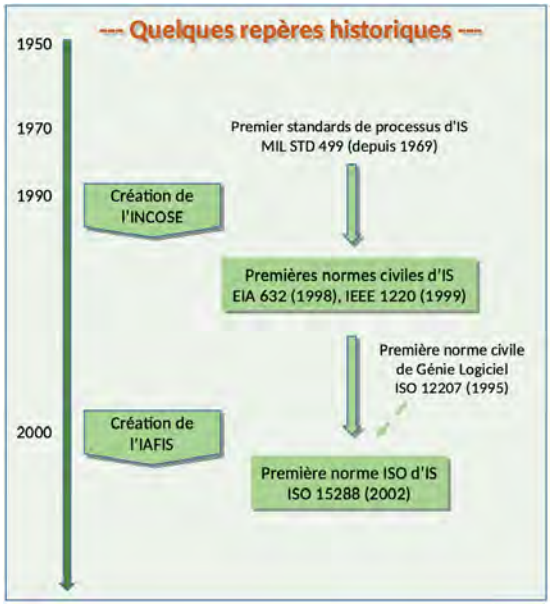

FIGURE 2.5 – Vue d'ensemble de l'arrivée de standards d'IS [Fio12].

Dans le cadre de cette thèse, seul le standard EIA-632 est retenu, car il prend en compte les parties descendante et montante du cycle en V. En effet, parmi les bénéfices potentiels de la proposition de simulation, il y a la possibilité, que les scénarios de vérification et validation puissent s'appliquer à l'ensemble de la partie montante de ce cycle en V. Ce standard est plus amplement abordé dans la section suivante.

2.6.1 Le standard EIA-632

Le standard EIA-632 [EIA99] (figure 2.6, [Fio12]), retenu dans le cadre de cette thèse, prend en compte : la définition du système, sa réalisation et son intégration, sans toutefois aller vers le cycle complet du produit, c'est-à-dire son exploitation et son retrait de service. Ces derniers aspects sortent du cadre de cette thèse, car la simulation y joue un rôle moins important, même si elle peut être utile dans le diagnostic opérationnel de pannes. Cette section présente une brève introduction de ce standard, et des éléments clefs qui le constituent.

2.6.1.1 Introduction

Ce standard a été conçu pour pouvoir s'appliquer à l'ensemble des secteurs et des domaines de l'industrie. Son objectif est d'aider à établir et à développer un ensemble d'exigences pour, in fine, fournir un système effectivement réalisable, dans une enveloppe

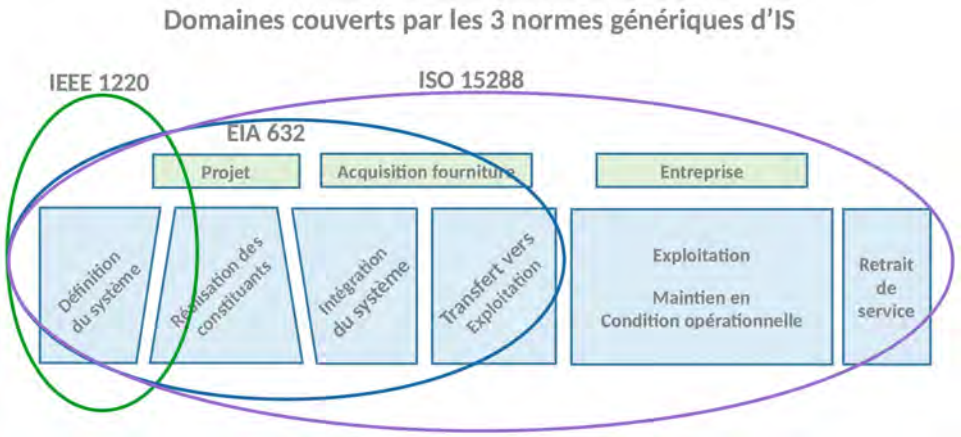

FIGURE 2.6 – Vue d'ensemble de l'étendue des standards d'IS [Fio12].

de coût, une planification et des risques déterminés. Conformément à l'esprit de l'IS, ce standard propose une approche systématique pour la création ou la réingénierie d'un système produit [EIA99]. En plus du système à créer, son autre intérêt, c'est sa capacité à prendre en compte les produits contributeurs, nécessaires à la réalisation du système. Par produits contributeurs, il faut comprendre, les outils de fabrication, de vérification, de qualification, et l'ensemble des logiciels associés.

2.6.1.2 Présentation

Avec ce standard, le système à réaliser est perçu comme une hiérarchie de sous-systèmes, auxquels, et pour chaque sous-système, les processus définis dans ce standard s'appliquent pleinement. Cela couvre les processus d'approvisionnement et d'acquisition des produits, de la gestion technique (le planning, l'évaluation et le contrôle), de la conception du système (les processus de définition des exigences et de la solution), de la réalisation du système (le processus de réalisation, et le processus d'installation), et de l'évaluation technique (les processus d'analyse du système, de vérification des exigences, de validation du système) [EIA99] (figure 2.7).

2.6.1.3 Éléments clefs

Le point clef de ce standard est le concept système (figure 2.8). Ce concept regroupe à la fois les produits à destination du client final (End Products), et l'ensemble des produits contributeurs (Enabling Products) pour développer (figure 2.9 : Development Products), réaliser (Production Products), tester (Test Products), déployer (Deployment Products) utiliser et assurer le support de ces produits. Il est à noter que ce concept inclut également et de

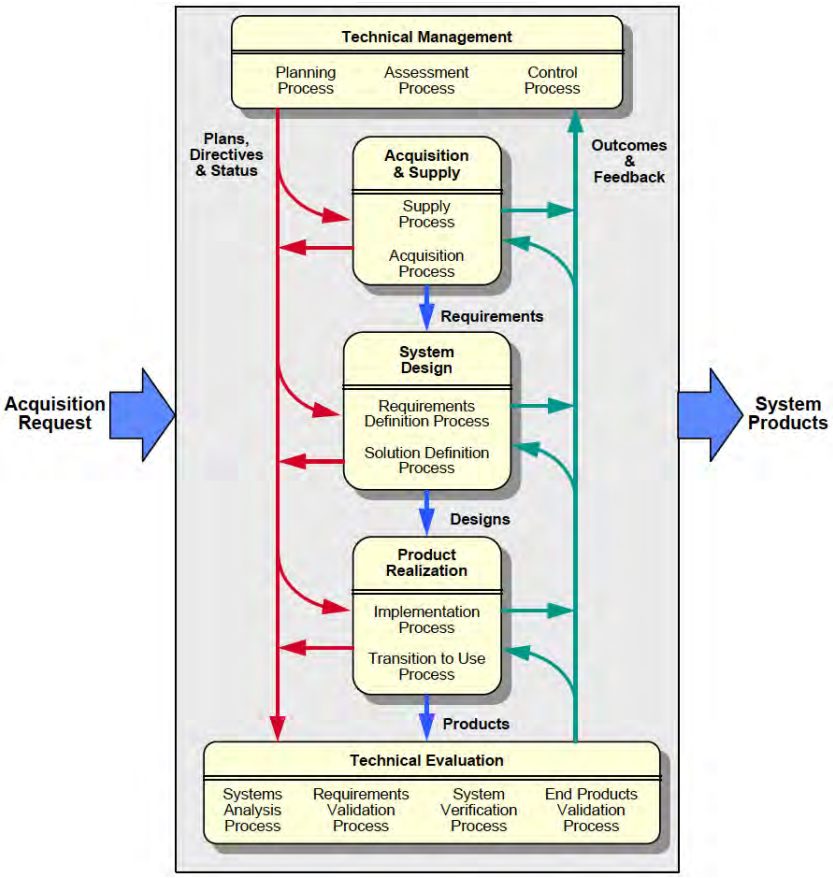

FIGURE 2.7 – Vue d'ensemble des processus de la norme EIA-632 [EIA99].

manière implicite l'ensemble des personnes impliquées dans le système.

Ce concept est décliné en Buiding Block tel qu'illustré par la figure 2.9. On y retrouve le System, le End Product et les Enabling Products Sets. Ce End Product est lui-même décomposé en Subsystem. Afin de développer ce Subsystem et par approche récursive, ce Subsystem devient lui-même un System avec son End Product et les Enabling Products sets associés.

L'application de ce concept de Buiding Block est plus large. Elle concerne également les équipes qui spécifient le système. Tout en gardant la structure de Buiding Block, les termes System, End Product, Subsystem, Development Products, deviennent dans ce cas System Specification, End Product Specification, Subsystem Specification, Development Products Specification, etc.

Le concept de Buiding Block prend en compte, également, le cas des équipes pluridisciplinaires. Ces équipes sont organisées en conformité avec cette structure Buiding Block de base. Dans ce cas et pour assurer une bonne identification, les dénominations deviennent

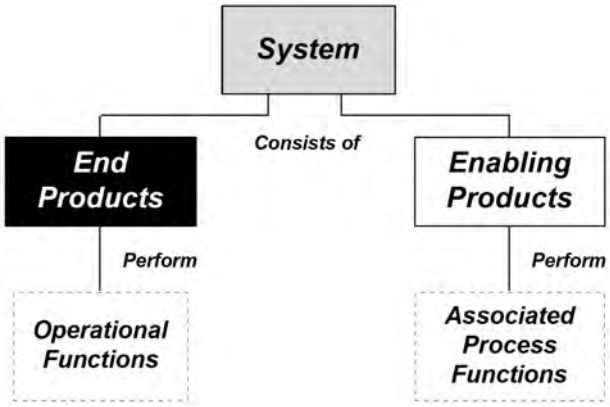

FIGURE 2.8 – Le concept système du standard EIA-632 [EIA99].

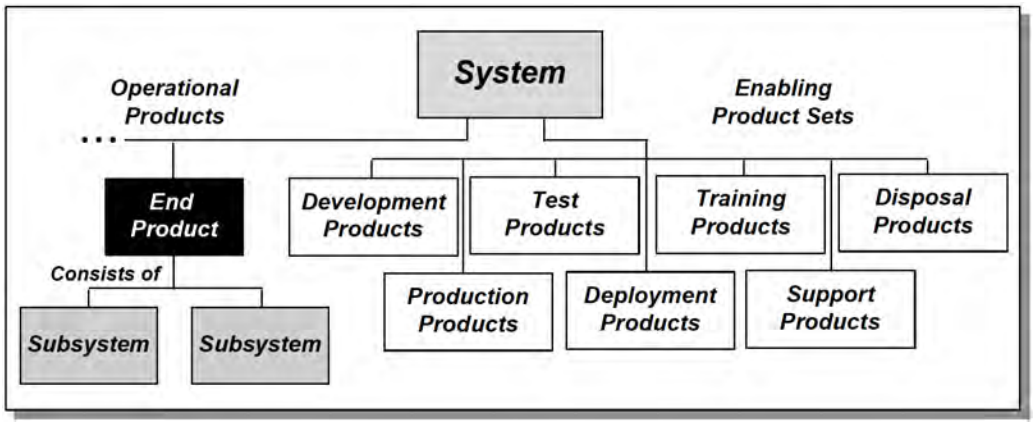

FIGURE 2.9 – Building Block [EIA99].

System Team, End Product Team, Subsystem Team, Development Products Team.

Ce standard précise les types de procédures à mettre en œuvre : de l'achat des fournitures à la mise en place du produit, puis à la formation des utilisateurs. Toutefois et d'après le standard EIA-632, c'est bien sous le terme Solution Defintion Process de l'étape System Design de la figure 2.7, que sont exécutées les procédures pour la prise en compte des exigences du client et pour l'analyse et la conception du produit. Cet endroit est donc le lieu approprié pour développer les analyses sous la forme d'ingénierie système dirigée par les modèles et pour la mise en œuvre de la simulation pour vérifier puis valider la conception du produit. Ce type d'ingénierie est développé dans la section suivante.

2.7 Ingénierie système dirigée par les modèles

2.7.1 Une vue simplifiée de l'IS d'hier et encore d'aujourd'hui

Longtemps, la mise en œuvre de l'IS (Ingénierie Système) a été réalisée sous forme papier ou avec sa déclinaison informatique, via des tableurs, traitements de texte ou logiciels de dessin. Les exigences sont décrites sous forme textuelle, voire avec des représentations graphiques non standardisées pour certains aspects structurels, voire dynamiques. Même si les objectifs de fiabilité, de réduction des coûts de conception, d'une meilleure définition des interactions entre les différents constituants d'un système (initialement définis par le DoD [10], puis transcrit via des normes civiles EIA-632, ISO-15288) peuvent être atteint, par la mise en œuvre de ces méthodes d'IS, il n'en reste pas moins que les modèles issus de ces analyses peuvent être considérés comme contemplatifs, dans le sens où les analyses de cohérence des exigences, des différents diagrammes entre eux, des différentes vues, dont la somme représente le système étudié, doit se faire à la main, sans réel support logiciel qui permettrait de formaliser puis mécaniser l'évaluation, la détermination de la complétude et de la cohérence, pour obtenir la confiance dans la conception du produit.

2.7.2 L'arrivée de l'ingénierie dirigée par les modèles

Les Grecs, puis bien plus tard les encyclopédistes ont cherché à classifier leurs connaissances du monde du vivant, sous la forme d'une taxonomie, par la distinction des caractéristiques entre les différentes espèces du règne animal. Dans les années 60 et en informatique apparaît de nouveau ce besoin de classification des connaissances. Pour ses besoins en intelligence artificielle (IA, 1965), Marvin L. Minsky [Min65] pose la base d'une abstraction, d'une représentation abstraite de la réalité sous la forme d'un modèle. Il en donne la définition suivante :

« *Pour un observateur* $\boldsymbol{B}$*, un objet* $\boldsymbol{A^*}$ *est un modèle d'un objet* $\boldsymbol{A}$ *dans la mesure où* $\boldsymbol{B}$ *peut utiliser* $\boldsymbol{A^*}$ *pour répondre aux questions qui l'intéressent au sujet de* $\boldsymbol{A}$*.* »

Dit autrement, le modèle est une représentation, une abstraction de la réalité, suivant un point de vue particulier. Il doit pouvoir être questionné pour en évaluer sa justesse, ou pour répondre à des questions sur cette partie de la réalité modélisée, pour la comprendre, l'expliquer, ou en prédire son évolution. Cela concerne aussi bien les modèles mathématiques formels, la physique, la biologie, les sciences humaines, ou encore tous autres domaines.

Toujours à cette même époque (1960-1970) et dans un domaine différent apparaît la

10. *Department of Defense (US)*

définition de la programmation par objet par leurs auteurs Ole-Johan Dahl et Kristen Nygaard (tous les deux prix Turing en 2001), puis développé par Alan Kay [Kay03] (1970, prix Turing en 2003). Alan Kay développe en 1971 le langage de programmation orienté objet : Smalltalk [Kay93], langage réflexif et dynamiquement typé. Ce langage introduit les notions d'objet, de classe, d'héritage, de polymorphisme et de métaclasse. Pour Alan Kay, l'objet peut être une cellule, une voiture ou tout autre objet au sens large. Le concept de modèle de Marvin L. Minsky et la classification des connaissances trouvent leur expression formelle dans cette programmation orientée objet, avec cette notion d'objet (représentation abstraite d'une réalité) et d'héritage (issu de la taxonomie).

Les liens entre les concepts de modèle et de programmation orientée objet sont en place. Vers les années 80, Grady Booch, Ivar Jacobson, James Rumbaugh créent séparément leur propre méthode, à base de représentation graphique, pour la conception et la modélisation de programme informatique orienté objet. Devant la diversité des représentations graphiques des objets, classes et de leurs interrelations, ils unifient leurs travaux pour répondre à une demande de l'OMG [11] et donner le langage UML [12] (Unified Modeling Language). Ce langage de modélisation UML 1.0 devient un standard adopté par l'Object Management Group (OMG) en janvier 1997.

Sur la base de ce langage UML et durant l'année 2000, l'OMG promeut le MDA (Model Driven Architecture) pour inciter aux bonnes pratiques de modélisation afin exploiter les avantages dus à l'emploi des modèles. Cette pratique s'appuie sur le langage UML, sur le MOF [13] et sur le standard d'enregistrement et d'échange structuré des données XMI [14]. Le MDA [OMG14] consiste en l'élaboration de modèles indépendants de toutes informatiques (CIM) (représentation du besoin métier), de modèles indépendants d'une plateforme d'exécution (PIM) (analyse et conception), puis en des modèles spécifiques à une plateforme d'exécution (le code sous-tendu par l'environnement d'exécution). Le but de ces trois distinctions : CIM, PIM et PSM est de permettre la génération automatique d'une partie des modèles et du code associé, pour obtenir un gain de productivité.

Un peu plus tard, en complément du MDA, et dans une vision plus large [Com08], l'approche IDM [15] (Ingénierie Dirigée par les Modèles) apparaît pour traduire les préoccupations des utilisateurs, des concepteurs et architectes, en lieu et place de la vision *objet*. Avec l'arrivée de cette IDM le tout est objet se transforme en tout est modèle, ou encore en tout est langage, dont la particularité est l'obtention de langages de modélisation spécifiques à un domaine métier (DSL). Le nombre de concepts métier mis en œuvre est moindre par rapport au langage UML.

11. *Object Management Group*
12. *Unified Modeling Language*
13. *Meta-Object Facility*
14. *XML Metadata Interchange*
15. *Ingénierie Dirigée par les Modèles*

De cette définition d'un modèle, il est possible d'en retirer 3 idées fortes : la notion de représentation, de justesse et d'exploitation. La première relation majeure de l'IDM est représentée par le concept de représentationDe. Comme l'exprime la définition, le modèle est une représentation d'une partie de la réalité. Pour la justesse et l'exploitation, [Com08] aborde le *Principe de substituabilité* dont la définition peut être donnée par : le modèle peut remplacer l'objet réel pour des besoins d'analyse. Il est alors dit descriptif de cette réalité. Ce modèle doit pouvoir être exploité et questionné tout en ayant la justesse suffisante. Les réponses attendues de ce modèle descriptif se doivent d'être conformes à ce que la partie de la réalité concernée par cette modélisation aurait pu répondre, pour un même questionnement. Mais lorsque le modèle précède la réalité, c'est-à-dire lorsque le modèle fait partie du processus initiateur du développement d'un système, alors ce modèle peut être défini comme prescriptif : le modèle devient la base des exigences, expression des besoins du client, pour le développement du système réel. Toutefois, cela pose la question de savoir quelles sont les conditions qui permettent de justifier la substitution du modèle prescriptif par le système réel, lorsque ce dernier est réalisé.

Une réponse peut être apportée par Barbara Liskov [LW96] dans son article « *A behabvioral Notion of Subtyping* », concernant la programmation orientée objet. Elle propose l'exigence de substituabilité suivante pour réaliser une substitution d'un objet super-type par un objet sous-type de ce même super-type :

> « *Subtype Requirement : Let ø(x) be a property provable about object x of type T. Then ø(y) should be true for objects y of type S where S is a subtype of T.* »

Dans une acception plus large que la programmation objet et qui semble pouvoir être appliquée à cette modélisation prescriptive, il est possible de définir, par l'analogie, les éléments super-type et sous-type. Le modèle prescriptif correspondrait au super-type « T ». Le système réel, développé à partir du modèle prescriptif, pourrait être considéré comme le sous-type « S » du super-type « T ». Typer ces éléments ne justifie toujours pas la possibilité de cette substitution du modèle prescriptif par le système réel.

Il est des cas, où même si les concepteurs se sont attachés à modéliser le mieux possible le système désiré, ce système dans la réalité peut faire apparaître des effets de bord non initialement prévus. Il est possible de citer, par exemple et suivant la nature du système : des couplages vibratoires, électromagnétiques, optiques, acoustiques, reflet de la réalité physique du système conçu. Puisque ces effets inopportuns n'ont pas été anticipés, le système ne peut pas se substituer au modèle prescriptif puisque certaines propriétés du système (les effets de bord) ne sont pas des propriétés prouvables du modèle prescriptif. Il y a alors deux alternatives dans ce processus d'ingénierie pour prendre en compte ces effets de bord : soit améliorer le système pour qu'il corresponde au modèle, soit réviser le modèle et les exigences associées (dans une approche hybride prescriptif/descriptif) pour que le modèle

corresponde au système. Avec cette dernière hypothèse, si les exigences sont précisées pour que le modèle prescriptif définisse un ensemble de propriétés (par exemple, les couplages) avec des intervalles de tolérance, le système réel pourra, alors, se substituer au modèle prescriptif, si l'ensemble des propriétés du système réalisé s'inscrit dans les intervalles de tolérance des propriétés du modèle prescriptif.

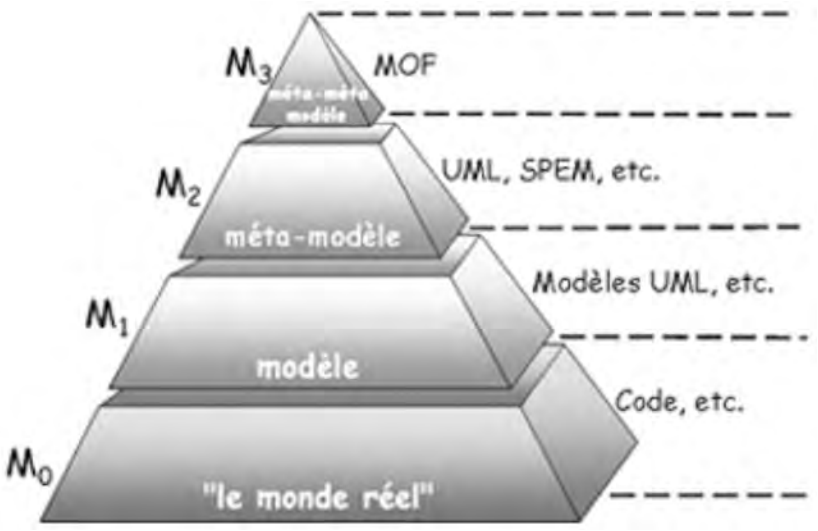

FIGURE 2.10 – La pyramide de modélisation [OMG16], extrait de [Com08].

Faire un modèle descriptif ou prescriptif (couche M1 de la figure 2.10) d'un objet réel (couche M0) nécessite l'emploi de symboles clairement définis pour exprimer les concepts nécessaires à la représentation de la partie de la réalité, centre de l'attention. Ces symboles textuels ou graphiques (sémantique concrète) peuvent être vus comme un langage (avec une sémantique abstraite sous-jacente), basé sur une grammaire ou encore un méta-modèle/méta-langage (couche M2 de la figure 2.10) [OMG16] [Com08]. Cela amène à la deuxième grande relation de l'IDM où le modèle doit être conformeA son méta-modèle. Dit autrement, le modèle doit être construit sur la base des concepts définis par le méta-modèle/méta-langage. Pour illustrer et en considérant une carte routière, la légende correspond à sa grammaire, à son méta-modèle. Tout ajout de symbole dans cette carte, ne faisant pas partie de cette légende, rend celle-ci non-conformeA sa légende.

Avec l'IDM, chacun a la possibilité de concevoir son propre méta-modèle, c'est-à-dire l'expression des concepts à mettre en œuvre pour modéliser son besoin métier. Devant la diversité grandissante des méta-modèles et de leurs potentielles incompatibilités, l'OMG a créé un langage commun de définition des méta-modèles, qui prend lui-même la forme d'un méta-modèle. Ce langage est dénommé *méta-méta-modèle* (couche M3 de la figure 2.10). Il possède la particularité d'être *circulaire*, c'est-à-dire se définissant lui-même.

L'OMG [OMG16] offre une représentation graphique de ces différents niveaux d'abs-

traction (figure 2.10), sous la forme d'une pyramide, où M0 représente le monde réel, dont une partie de celui-ci peut être modélisée suivant un point de vue d'intérêt, M1 est bien le niveau du modèle, représentation abstraite et partielle de ce monde réel M0, M2 le niveau du méta-modèle, du langage qui permet de décrire le niveau M1, enfin le pyramidion (le niveau M3 méta-méta-modèle, le MOF dans le cadre de l'OMG, ou Ecore [16], dans le cadre de l'environnement EMF [17], fournit les concepts, le langage pour concevoir les méta-modèles.

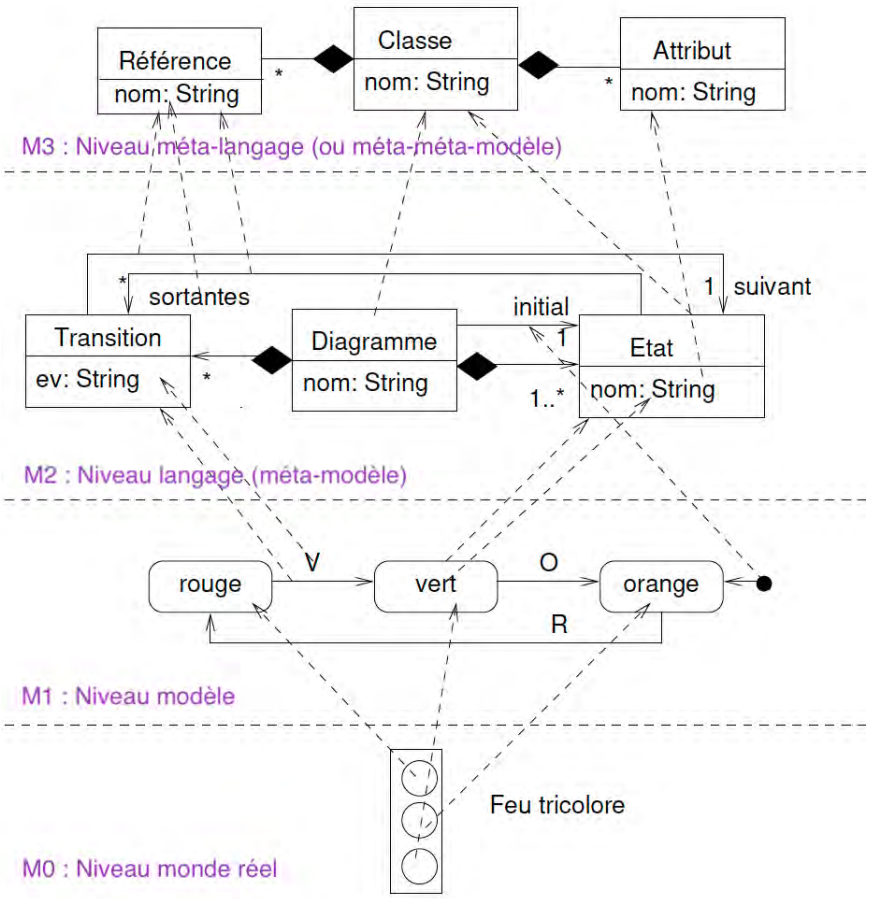

FIGURE 2.11 – Instanciation de la pyramide de l'OMG pour un feu tricolore, [Cré+13].

Un exemple de l'instantiation de la pyramide de l'OMG est donné en figure 2.11 pour un feu tricolore [Cré+13]. Comme dans la définition de la pyramide de l'OMG, le niveau M3 contient les méta-méta-modèles de base que sont, dans l'exemple, les méta-méta-classes *Classe*, *Attribut Référence* et les liens de composition qui les unissent. Elles permettent de se décrire elles-mêmes : par exemple, la méta-méta-classe *Classe* possède un attribut (nom : String) qui fait référence à la méta-méta-classe *Attribut* et deux liens de composition vers les deux autres méta-méta-classes *Référence* et *Attribut* qui font également référence à la

16. *Eclipse Core*
17. *Eclipse Modeling Framework*

méta-méta-classe *Référence*. Ce niveau M3 est le pyramidion de la pyramide de l'OMG, et paradoxalement, c'est la pierre angulaire qui va permettre de construire les langages comme UML ou des langages spécifiques à des métiers (DSL) au niveau de la couche M2. Ce niveau M2 décrit la structure métier du modèle, dont il sera possible d'en trouver une instanciation dans la couche M1. Dans l'exemple de la figure 2.11, la structure du feu tricolore peut être modélisée par des objets conformes aux méta-classes *Diagramme*, *Transition* et *Etat*. Le méta-modèle de cette couche M2 est conforme au méta-méta-modèle (couche M3) suivant deux conditions : l'ensemble des éléments de cette couche M2 sont des instances des éléments de la couche M3, et les contraintes exprimées sur le méta-méta-modèle de la couche M3 sont respectées au niveau de la couche M2 [Cré+13]. Dans l'exemple, ce modèle est une machine à états, dont les états (instance de la méta-classe *Etat*) sont nommés *rouge*, *vert* et *orange*. Ces mêmes états sont reliés entre eux par des transitions instance de la méta-classe *Transition*. Ce modèle décrit le comportement attendu du feu tricolore dans le monde réel (couche M0). Dans cet exemple et dans le monde réel, le feu tricolore passera successivement et indéfiniment de la couleur *rouge*, à la couleur *verte* puis à la couleur *orange*.

Comme montré dans l'exemple précédent, l'IDM fait explicitement référence à des langages bien définis pour concevoir les modèles. Sur cette base, le paradigme tout est modèle peut aisément être modifié en tout est langage. Or, le grand avantage de l'emploi des langages est de permettre l'automatisation de tâches, ou encore dans le cas de cette section, de rendre opérationnels des modèles pour une simulation. Parmi ces attraits, il en est un particulièrement important, c'est la possibilité de pouvoir transformer des modèles, pour pouvoir passer d'une représentation métier à une autre, en définissant un langage (figure 2.12-MMt) et un modèle de transformation (figure 2.12-Mt). La transformation au sens large peut être *exogène* ou *endogène*. Par *exogène*, il faut comprendre que les méta-méta-modèles (niveau M3) ou les méta-modèles (niveau M2) d'entrées (MMa) et de sorties (MMb) de la transformation sont différents, tandis que dans les transformations *endogène*, les méta-modèles MMa et MMb sont identiques.

L'intérêt de cette transformation de modèles est de pouvoir faire de la génération de code, de la migration technologique, ou encore (et sans être limitatif) de la restructuration de modèle ou de code.

Cette vision du tout est langage donne à chacun la possibilité de développer sa propre grammaire, pour représenter, au plus juste, les concepts prévalant au sein de son propre métier. Or devant la diversité des langages (DSL) mis en oeuvre pour réaliser des systèmes compliqués, voire devenant complexe, les données et le sens attaché à celles-ci, leurs traductions dans les langages cibles peut devenir une activité particulièrement ardue, lorsque les langages sources et cibles sont dits *exogènes*, ou encore, lorsque les concepts métier du langage source (MMa) sont difficilement traduisible dans le langage cible (MMb).

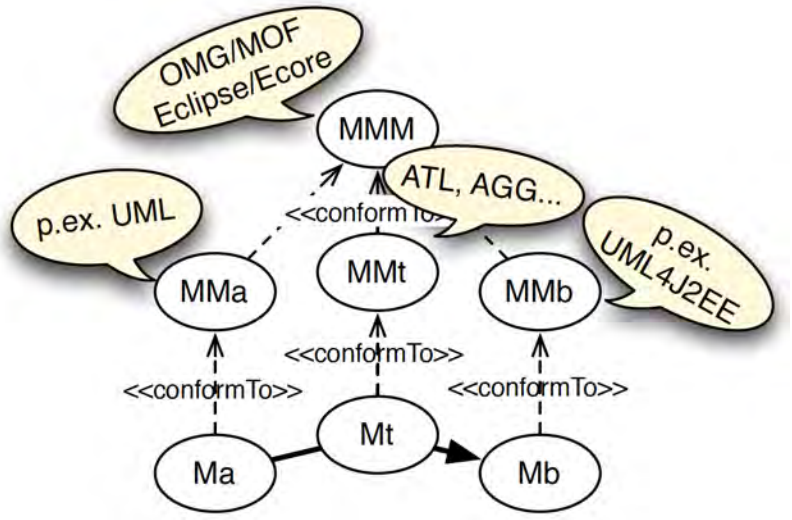

FIGURE 2.12 – Principe de la transformation de modèles, extrait de [Com08].

À cette difficulté de traduction, le MOF impose des restrictions telles qu'un modèle ne peut être une instance que d'un seul méta-modèle, ou encore l'interdiction d'agréger des niveaux différents : modèle-méta-modèle [Kou+13] dans un même espace technologique. Or il arrive que l'intersection de deux langages mette en exergue un même concept, interdisant, de facto, leur mise en relation dans un même espace technologique, du fait de cette instanciation unique.

Pour palier aux restrictions du MOF, [Gol+16] et [Kou+13] proposent une autre approche basée sur la notion de fédération de modèles. L'objectif, avec cette fédération, est de conserver les modèles dans leur propre espace technologique (cf. Espace d'information figure 2.13), tout en fournissant un mécanisme pour que les éléments de ces différents modèles technologiques puissent être accessibles, pour développer des modèles, des vues transverses (Espace conceptuel), pouvant répondre aux demandes des concepteurs (Espace de conception).

Ces modèles et vues créés dans l'*espace conceptuel* sont en fait des modèles virtuels. Le terme virtuel signifie que les modèles regroupent et réutilisent des fonctionnalités déjà disponibles dans chaque espace technologique. La difficulté de cette approche fédérée repose sur les connecteurs (carré noir, figure 2.13) qui font le lien entre l'*espace conceptuel* et les *Espaces de conception* et *Espace d'information*. Ces connecteurs doivent permettre la création de liens dynamiques [Gol+16], qui nécessitent parfois des traductions sémantiques complexes [Kou+13], entre les modèles virtuels de l'*espace conceptuel* et les espaces environnants. À cette difficulté initiale, il est également nécessaire d'ajouter la maintenance en condition opérationnelle de ces connecteurs, en fonction des évolutions des outillages des espaces technologiques comme de l'espace de conception. L'autre difficulté, c'est l'absence

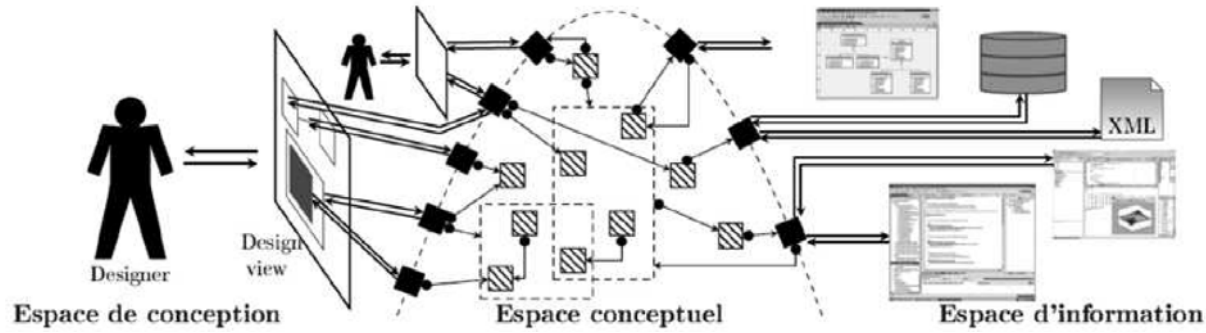

FIGURE 2.13 – Illustration de cette fédération de modèles pour la conception de systèmes complexes hétérogènes [Kou+13].

potentielle de connecteurs disponibles pour un espace technologique spécifique. Ce concept de fédération de modèles et de sa méthodologie associée ont donné lieu au développement d'un outil open source *Openflexo* [Ope19]. Une mise en œuvre de ce concept de fédération a été réalisé sur un cas d'étude concernant une collectivité territoriale [Beu+15].

C'est bien l'ensemble des concepts liés à cette IDM qui, appliqué à l'ingénierie système, donne toute la force à l'ingénierie système dirigée par les modèles. La section suivante illustre la force de cette nouvelle ingénierie.

2.7.3 Intérêt de l'ingénierie système dirigée par les modèles

L'ingénierie système dirigée par les modèles (MBSE [18] en anglais) peut être considérée comme l'alliance de la puissance méthodologique et d'organisation apportée par l'ingénierie système et de la puissance productive de l'Ingénierie Dirigée par les Modèles (IDM). Cette alliance décuple les possibilités offertes aux concepteurs. En effet, lorsque les logiciels intègrent ces deux composantes, l'utilisateur dispose des outils pour rendre ses modèles plus productifs (basés sur des langages formels : informatisés, mathématisés). Cet aspect productif s'exprime par les possibilités :
a) d'avoir des fonctions initialement définies dans une vue, qui puissent être réutilisées tout en conservant leurs propriétés, dans d'autres vues,
b) d'offrir un contrôle de cohérence entre les différents diagrammes, par exemple entre les diagrammes de séquence et d'activité,
c) de pouvoir mettre en place des liens entre les différentes couches systèmes et d'y pouvoir faire des analyses de traçabilité,
d) d'y ajouter des points de vues, avec le langage de définition (méta-modèle) propre à chaque métier,

18. *Model-Based Systems Engineering*

e) de faire, au plus tôt (early), de la vérification et validation structurelle, voire comportementale, si la sémantique des modèles le permet,
f) de faire des transformations de modèles,
g) et toutes autres nécessités qui puissent être réalisées via un langage de modélisation.

Parmi ces possibilités diverses et variées, le premier groupe de travail du projet MOISE [19] a pu, avec l'aide du MBSE, réaliser et démontrer la pertinence de mettre en place un processus d'amélioration des exigences du client. Le deuxième groupe de travail a défini un processus de synchronisation entre les modèles d'analyse fonctionnelle et les modèles d'analyse de la sûreté/sécurité. Issu de cette réflexion, le tout nouveau projet S2C [20] tend, entre autres, à mobiliser les puissances de l'IS et l'expressivité de l'IDM pour concrètement mettre en œuvre ce processus de synchronisation. Le recours à l'IDM a été indispensable pour le quatrième groupe de travail, puisqu'il a pu produire une preuve de concept pour agréger les modèles système et métiers des différents partenaires d'une entreprise étendue, définis en utilisant différents langages et outils de MBSE, pour obtenir une vue d'ensemble du système. Enfin, le troisième groupe, en s'appuyant sur le MBSE a pu démontrer concrètement la mise en œuvre de cette proposition de thèse, via la définition de nouveaux méta-modèles. Il a ainsi montré la faisabilité de la vérification/validation, au plus tôt, des aspects dynamiques du modèle système étudié, via la transformation de modèle qu'est la génération de code. Sur la base du modèle système de la couche physique, cette génération de code a permis de créer automatiquement la liste des interactions avec le bus de cosimulation et de définir les API des fonctions pour la simulation. Quelques standards de simulation sont abordés dans la section 2.8, page 40.

2.7.4 La méthode d'ingénierie système dirigée par les modèles : RFLP

Il existe plusieurs méthodes pour définir, concevoir un système en s'appuyant sur l'Ingénierie Dirigée par les Modèles. L'une d'elles, la méthode RFLP se définit par la mise en œuvre de 4 couches d'analyses : « Requirements, Functional, Logical, Physical » [KK13] [Les19].

La première couche, la couche *requirements* contient l'ensemble des exigences associées au produit à réaliser. Elle est l'expression du besoin du client. Cette couche précise les services que doit rendre le produit vis-à-vis de son environnement sous la forme d'une description des interrelations avec les autres systèmes ou avec les acteurs devant le manipuler. Cette couche peut être également dénommée *operational*. Dans ce cas, elle ajoute, aux exigences définies précédemment, la notion de *missions opérationnelles*, pour lesquelles il

19. *MOdels and Information Sharing in Extended Enterprise*
20. *System & Safety Continuity*

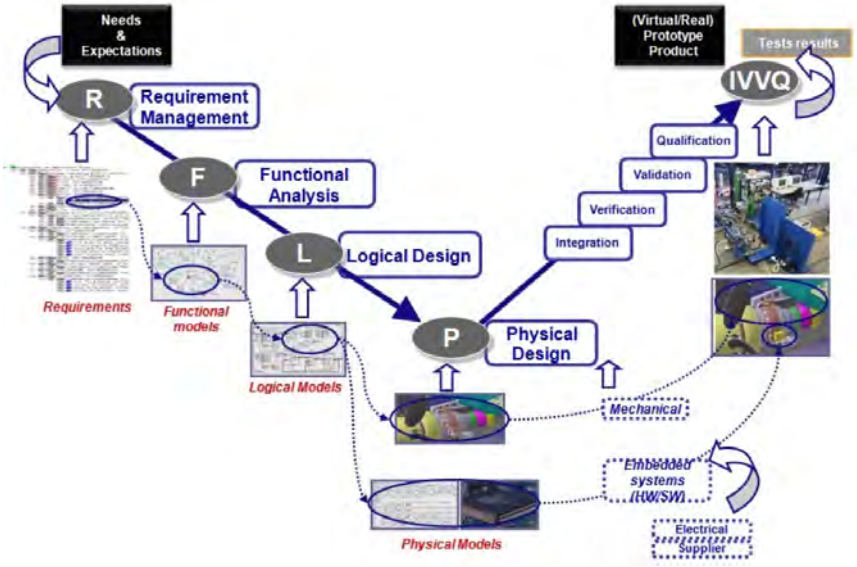

FIGURE 2.14 – Méthode RFLP [Les19].

faut définir les services que le système doit rendre à son environnement pour remplir ses missions [Fio12].

C'est dans la couche *fonctionnelle* que l'ingénieur système crée son architecture fonctionnelle. C'est-à-dire qu'il définit les fonctions de premier niveau qui doivent être mises en œuvre, puis il les raffine jusqu'à obtenir une architecture répondant aux exigences décrites dans la couche précédente, tout en mettant en évidence les interrelations entre les fonctions feuilles.

La couche *physique* précise l'architecture physique du produit. Elle décrit les constituants physiques, les équipements, les liaisons mécaniques, électriques et autres, sur lesquels les fonctions sont allouées.

La couche *logique* est une couche intermédiaire entre la couche fonctionnelle et la couche physique. Cette couche a pour vocation de réorganiser la couche fonctionnelle en regroupant les fonctions feuilles suivant les flux échangés et transformés (énergie, information), les flux de contrôle (informations de déclenchement et régulation des fonctions). Cette réorganisation suppose de prendre pleinement en compte l'ensemble des modes de fonctionnement, des missions [Fio12]. Dit autrement, le regroupement des fonctions peut se fait sur la base des différents corps métiers (mécanique, électrique, etc.) en prévision de leur affectation sur les équipements physiques.

2.7.5 La méthode d'ingénierie système dirigée par les modèles : CESAM

Différemment de la méthode RFLP, la méthode CESAM propose une architecture en 3 couches (figure 2.15) : les couches *opérationnelle, fonctionnelle et de construction.*
La couche *opérationnelle* précise l'environnement et définit les interactions et les interfaces

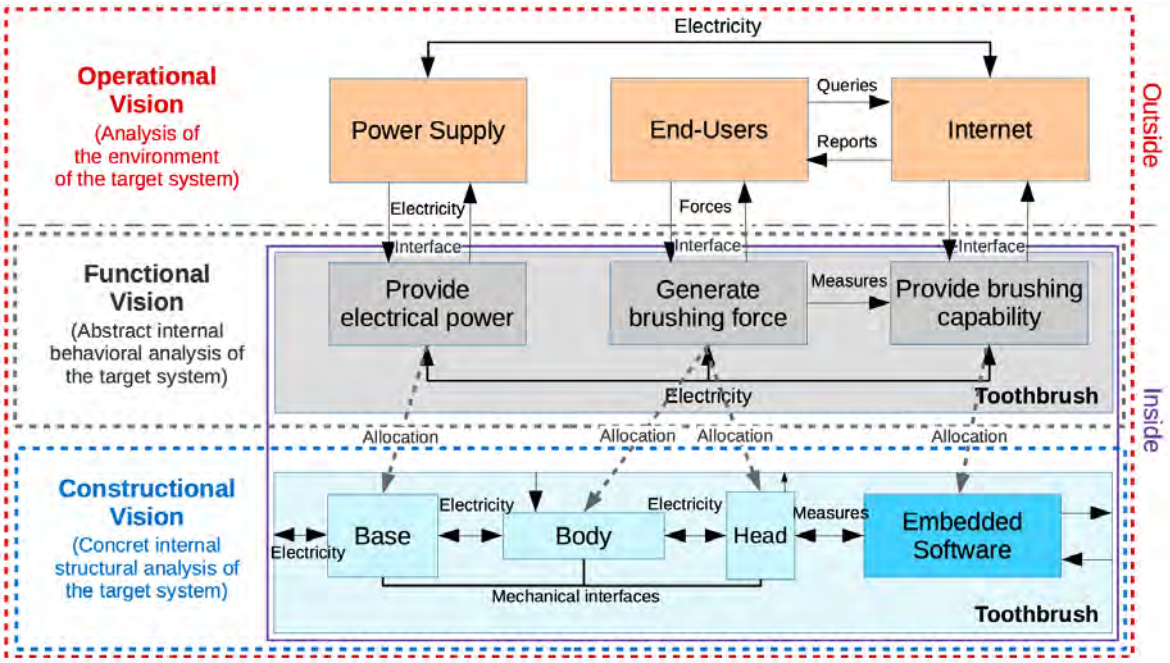

FIGURE 2.15 – CESAM, exemple d'une vision architecturale [CES19].

avec le système, vu comme une boîte noire. La couche *fonctionnelle*, à l'instar de la méthode RFLP, est dédiée à l'analyse comportementale, sans faire référence à des éléments concrets du système. Cette couche peut être raffinée jusqu'à obtention d'une architecture respectant les exigences de besoins. La dernière couche est dite de *construction.* Elle décrit la structure physique du système, avec les différentes liaisons mécaniques, électriques, et autres. Chacun des composants physiques se voit allouer des fonctions issues de la couche supérieure.

2.7.6 La méthode d'ingénierie système dirigée par les modèles : ARCADIA

Cette méthode *ARCADIA* [21] a été initialement développée, au sein de la société Thales, par et pour les ingénieurs systèmes et de spécialités [Voi18]. Elle est, depuis, largement dé-

21. *ARChitecture Analysis and Design Integrated Approach*

ployée au sein de cette société. Cette méthode est une méthode structurée pour vérifier et valider des architectures système. Elle s'applique à des domaines aussi variés que l'aéronautique, la construction navale, le ferroviaire, le spatial, et tous autres secteurs nécessitant de l'ingénierie. Elle favorise la collaboration des différents corps de métier pouvant intervenir dans la définition du système à réaliser (la sécurité, la sûreté, la mécanique, la performance, les coûts, etc.) [Roq18]. Le grand avantage de cette méthode est de fournir aux différents intervenants, la même méthode, les mêmes outils, les mêmes informations sous la forme de modèles partagés.

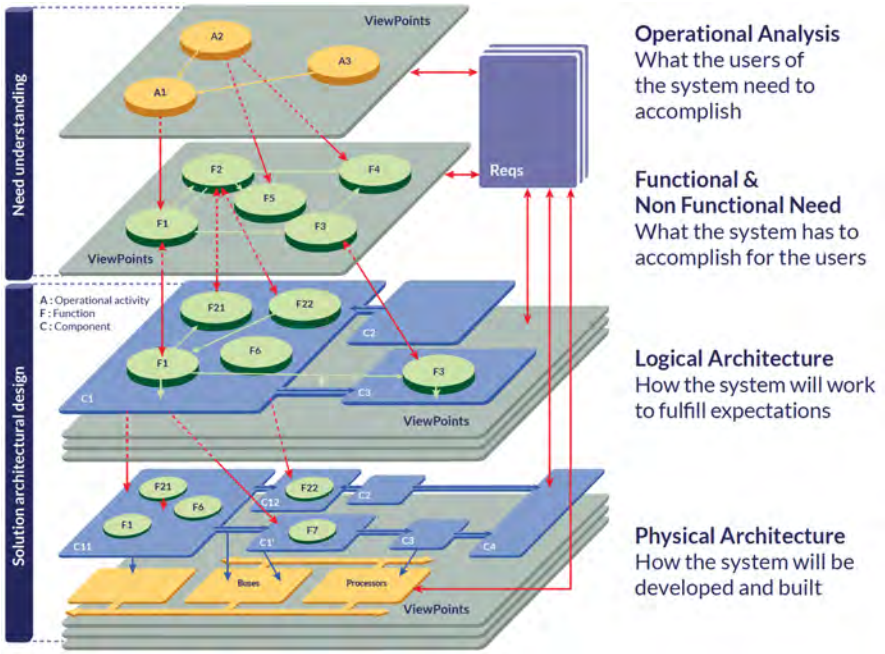

FIGURE 2.16 – Les 4 principales couches systèmes de la méthode ARCADIA [Cla19].

Cette méthode s'étend de la phase de capture des besoins opérationnels du client jusqu'à la définition de l'architecture physique du système (figure 2.16).

L'objectif de la couche *opérationnelle* de la méthode *ARCADIA* est de modéliser les acteurs et leurs besoins opérationnels. En d'autres termes, cette modélisation décrit ce que doivent accomplir les acteurs (systèmes ou personnes) pour faire fonctionner le système. À ce stade, le système n'est pas encore identifié en tant qu'élément de modélisation [Roq18]. La couche *fonctionnelle*, quant à elle, est en miroir de la couche précédente. Elle consiste à identifier ce que le système doit faire pour répondre aux attentes opérationnelles des acteurs. Cette couche permet de faire une analyse fonctionnelle externe, puisque le système, à ce niveau, est encore perçu comme une boîte noire. La couche *logique* est dédiée à l'analyse fonctionnelle interne. Cette analyse consiste à raffiner les fonctions identifiées dans la couche précédente, à les regrouper en des composants logiques (généralement suivant les différents corps de métiers). Cette couche précise la manière dont le système devra travailler pour

répondre aux attentes opérationnelles des acteurs du système. Enfin, le couche *physique* définit l'architecture finale du système, telle qu'elle doit être réalisée [Roq18]. Cette couche contient les différents éléments physiques (calculateur, serveur de données, moteurs, etc.), auxquelles sont affectés les différents composants logiques (et leurs fonctions) pour réaliser le comportement attendu du système.

2.7.7 Le logiciel Capella

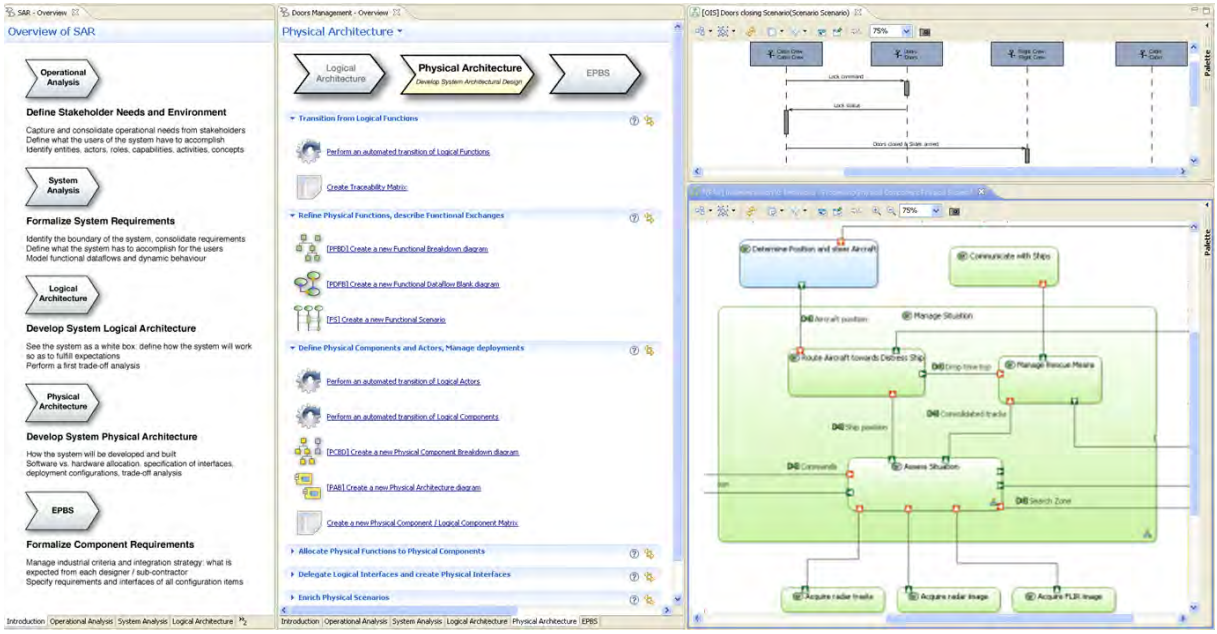

FIGURE 2.17 – Vue d'ensemble de Capella [CAP19].

Le logiciel *Capella* a été créé, concomitamment à la méthode *ARCADIA*, au sein de la société Thales. Il intègre la méthode *ARCADIA* et bénéficie des retours d'expériences des ingénieurs système et spécialistes quant à son utilisabilité et aux outils nécessaires pour rendre leur travail efficace. Ce logiciel est maintenant placé en *Open Source* dans le but de faire éclore un écosystème [Roq18].

Les points importants, d'après [Roq18], sont l'intégration d'un guide méthodologique sous la forme d'un explorateur (Activity Explorer) (figure 2.17), un explorateur sémantique qui permet, pour un objet donné, de visualiser (référencé par ou référençant) les autres objets en lien direct amont ou aval, l'intégration de la validation des règles méthodologiques d'*ARCADIA*, la transition entre les différentes couches des éléments de modélisation des couches précédentes, la possibilité de créer des sous-systèmes à partir d'éléments de la couche physique, une grille de couleurs sémantiques (des couleurs dédiées à chaque niveau

d'analyse : en vert pour les fonctions, en bleu pour les composants, en rose pour les interfaces et les modèles de données, en jaune pour les éléments physiques), un *Diff/Merge* efficace et dédié aux modèles.

Cette section a décrit, succinctement, trois méthodes d'ingénierie système dirigée par les modèles. Si les trois méthodes ont le même objectif : concevoir des systèmes répondant aux exigences du client, elles présentent des différences dans la manière d'aborder le développement d'un système. La méthode CESAM s'étend sur trois couches tandis que les méthodes RFLP et ARCADIA proposent quatre couches. Ces deux dernières méthodes diffèrent également par la sémantique de leurs couches d'analyse. La première couche n'a pas exactement le même sens pour ces deux méthodes. Dans la méthode RFLP, cette couche décrit les exigences et les interactions associées au système. Le système est alors perçu comme une boite noire. Dans la méthode ARCADIA, cette première couche dite *opérationnelle* décrit les missions des utilisateurs, sans aborder le système. Pour la couche fonctionnelle, dans la méthode RFLP, le système est perçu comme une boite blanche, avec une description des fonctionnalités internes au système. Pour la méthode ARCADIA, cette couche fonctionnelle décrit les services que doit rendre le système aux utilisateurs. Quant à la troisième couche dite *logique*, elle décrit les fonctionnalités internes du système (boîte blanche) pour la méthode ARCADIA et sert également à regrouper les fonctionnalités en des composants logiques pour ces deux méthodes RFLP et ARCADIA. Enfin, la couche physique, quelle que soit la méthode, décrit l'architecture finale du système.

La proposition méthodologique de thèse (chapitre 3) permet de prendre en compte la diversité (source de richesse) des approches proposées par ces méthodes d'ingénierie système. Les deux sections suivantes présentent des méthodes et outils centrés sur la modélisation comportementale des fonctions système et sur la simulation.

2.8 Cosimulation

2.8.1 Introduction

L'ingénierie système dirigée par les modèles (MBSE en anglais), la vérification et la validation au plus tôt (early V&V : *Vérification & Validation*) sont maintenant des éléments clefs pour le développement de système cyber-physique complexe (CPS [22]) dans beaucoup de domaines d'application, comme par exemple dans le domaine des transports. Dit autrement, les modèles, les modèles de simulation, la conception dirigée par la simulation sont des approches efficaces pour évaluer, vérifier & valider, très tôt dans le cycle de vie du produit, les décisions de conception par rapport aux besoins du client. La du-

22. *Cyber-Physical System*

rée et les coûts de développement du système peuvent en être ainsi fortement réduits [Fio12][BKC17][TH13][She+04], tout en améliorant la qualité du système.

Étant donné que les systèmes actuels deviennent de plus en plus compliqués, [Gra+13] propose une méthode MBSE ([INC15]) pour intégrer les activités de simulation dans le processus de développement système. En fonction du point de vue à évaluer, les activités de simulation permettent d'estimer et/ou d'affiner les différentes interactions entre divers composants [She+04][Gra+13], et dans des domaines métiers aussi divers que la mécanique, l'hydraulique, l'électrique, etc..

Tout en visant la réduction des temps et des coûts de développement, [Sir+15] vise les problèmes d'intégration, de vérification, de validation et de qualification des modèles de simulation. Pour réduire les potentielles ambiguïtés entre les ingénieurs système et les experts en charge du développement des modèles de simulation, [Sir+15] ajoute, dans ce processus, un nouvel acteur dénommé *Architecte des modèles* qui coordonne les diverses activités destinées à la création de modèles de simulation. Le nouvel acteur devrait avoir une vision multidisciplinaire lors de la conception de l'architecture du produit, et quelques connaissances des technologies de simulation utilisées pour modéliser les différentes parties de l'architecture.

Pour améliorer la communication entre l'architecte système et les experts en charge du développement des modèles, [Sir+15] propose une ontologie dénommée MIC [23], pour spécifier la partie structurelle des modèles de simulation et de leurs interfaces (figure 2.18). Ce MIC contient les éléments structurants de communication entre les différents intervenants pour réaliser les modèles de simulation. Ce MIC prend également en compte les besoins de divers domaines d'application et cible tous les acteurs gravitant autour des modèles.

En complément de ce MIC, et pour définir pleinement les modèles de simulation, [Sir+15][Ret+14][Ret+13] ajoute le concept de Modèle d'intention (MoI [24]). Ce MoI est conçu sur une approche, basée modèle, et cherche à spécifier le comportement des modèles de simulation, pour un scénario de simulation donné. Cette spécification peut se faire à l'aide de logiciel de modélisation tels que Modelica, Simulink, Java, C++ ou tous autres logiciels permettant de décrire le comportement d'un modèle destiné à la simulation. La combinaison du MIC et du MoI permet de combler le fossé existant entre les exigences de la simulation, spécifiées par l'architecte du système, et la mise en œuvre de tous les modèles de simulation requis pour une simulation donnée. Cette approche tend à réduire les problèmes liés à l'intégration de ces modèles de simulation.

23. *Model Identity Card*
24. *Model of Intention*

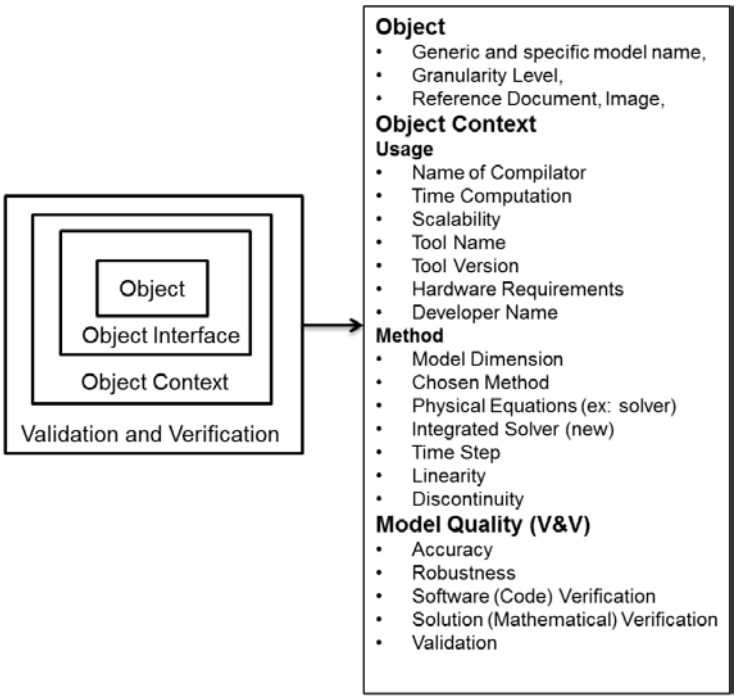

FIGURE 2.18 – Vue partielle du MIC, sans la partie interface [Sir15].

2.8.2 Quelques types de simulations

Il existe différentes catégories de simulation, qu'il est possible de classer de la manière suivante [DSC12] :
a) des applications de simulation numérique (appelées également simulations constructives),
b) des simulateurs pilotés (avec Homme dans la boucle mettant en œuvre du matériel simulé en totalité, généralement temps réel (appelés simulation virtuelle dans le jargon actuel),
c) des simulations vivantes et/ou instrumentées (exercices ou expériences avec des Hommes dans la boucle mettant en œuvre du matériel réel (Hardware in the Loop) ou en partie simulé, évidemment temps réel).

À cette catégorisation, il est possible d'ajouter qu'un système de simulation peut être implanté par un ensemble de modèles interagissant sur un réseau (local ou distant). Ce système de simulation peut alors être qualifié de système de simulation réparti ou encore de système de simulation coopératif et distribué. À ces types de simulations s'ajoutent des outils divers, généralement logiciels, pour gérer le système et ses données, visualiser, superviser, gérer les contraintes de sécurité et de protection de la propriété intellectuelle.

Les simulations peuvent être de type - discret, dans le sens où c'est l'arrivée des événements qui pilote l'avancement de la simulation - continu lorsque le temps évolue continûment et que la simulation est basée sur des équations différentielles ou d'intégration -

hybride si la simulation contient des modèles de type discret qui sont combinés avec des modèles de type continu.

La simulation répartie interagit avec différents modèles via le réseau de communication. La notion de réseau peut être, ici, prise au sens large, dans la mesure ou la communication entre les modèles peut se faire via un processus d'inter- communication IPC [25] lorsque les modèles sont exécutés sur des calculateurs mono ou multi-cœurs, ou encore via un processus d'intercommunication par le réseau IPNC [26], comme dans le cas de grilles de calcul.

Toutefois, que les modèles puissent être discrets, continus ou hybrides, que la communication se fasse par IPC ou IPNC, il n'en reste pas moins qu'il faille définir les interfaces de communication, et les processus d'échange des données entre les différents modèles, pour assurer leur interopérabilité, et maîtriser les contraintes en débit, latence, gigue du réseau et son impact sur la qualité de la simulation. C'est l'objet de la définition de standards de simulation dont, par exemple, les standards HLA et FMI. Ces deux standards sont abordés dans les sous-sections 2.8.4 et 2.8.5.

2.8.3 Différentes gestions du temps

Pour réaliser des simulations distribuées et déterministes, [Fuj98] définit différentes notions de temps et propose d'employer l'horodatage des messages. Trois types de temps sont définis : a) le temps physique du modèle simulé, c'est-à-dire le temps pour lequel l'on souhaite observer l'évolution du modèle. Par exemple, l'évolution du modèle entre les dates de 14 juin 2019 à 7H00 jusqu'au 15 juin 21H00, b) le temps de la simulation, plus exactement la représentation du temps dans la simulation : son étendue et son unité de temps. Par rapport à l'exemple précédent, cela donne une étendue de [0.00 .. 38.00] heures, pour une unité de temps de 1,0 heure, avec des nombres représentés en flottant, c) le temps physique réel de l'horloge murale qui représente notre écoulement du temps. Une seconde de temps écoulé représente alors bien une seconde de temps au sens de la définition des poids et mesures du système international.

Dans une simulation distribuée, horodater les messages permet de s'affranchir des variations des vitesses de transfert du réseau et de la difficulté de synchroniser les horloges de chaque calculateur (notion d'horloge logique). De cette manière, s'il est possible de recevoir un message indiquant une action réalisée, avant d'avoir reçu le message déclenchant cette action, horodater les messages permet de s'assurer de la causalité des messages.

25. *Inter-Process Communication*
26. *Inter-Process Network Communication*

2.8.4 Le standard de simulation HLA

HLA (High Level Architecture) est un standard qui fournit des spécifications pour réaliser des simulations coopératives et distribuées, avec l'objectif de faciliter la réutisabilité, l'interopérabilité et la composabilité des simulations [IEE10a].

2.8.4.1 Un peu d'histoire

Ce standard résulte de travaux menés par le ministère de la défense des USA (US Defense community) depuis la fin des années 70 jusqu'à la fin des années 80, pour créer et développer une technologie de simulation distribuée : le projet SIMNET [27] (figure 2.19). Suite à ce projet, le protocole de simulation interactive DIS [28] est devenu un standard industriel IEEE en 1993 : IEEE 1278 Distributed Interactive Simulation Standard Protocol. Ce standard définit les messages et les procédures pour la communication entre les simulateurs [Oka+16]. Ce standard IEEE 1278 a été retiré en faveur du standard HLA en 1998, et définitivement supprimé en 2010.

Au cours de l'année 1990, la DARPA [29] parraine une étude pour agréger des simulations distribuées entre elles : Aggregate Level Simulation Protocol ALSP [30]. Ce protocole intègre la gestion des données, la gestion du temps et la distribution des événements entre ces différentes simulations distribuées [Oka+16].

Les retours d'expériences et des concepts développés pour DIS comme pour ALSP sont à l'origine de la définition d'un cadre de technologies communes pour le développement de simulations distribuées. Il est formalisé par le standard HLA version 1.3 en 1998 [Oka+16] ; [DFW97a] ; [DFW98]. HLA devient un standard IEEE en 2000, nommé "Modeling & Simulation (M&S) High Level Architecture IEEE 1516-2000 series (IEEE 2000a, b, c)" [Oka+16]. En 2010, une nouvelle version de HLA est publiée et prend en compte les retours d'expériences des industriels : IEEE 1516-2010 series (IEEE 2010a, b, c) [Oka+16].

2.8.4.2 Quelques fondamentaux de HLA

Le standard HLA est défini par 3 livrets [IEE10a], [IEE10b], [IEE10c]. Le livret HLA Framework and Rules Specification [IEE10a] spécifie les éléments de conception d'un système et introduit les règles pour la conception de systèmes de simulation distribuée [Oka+16].

27. *Simulation Networking*
28. *Distributed Interactive Simulation*
29. *Defense Advanced Research Projects Agency*
30. *Aggregate Level Simulation Protocol*

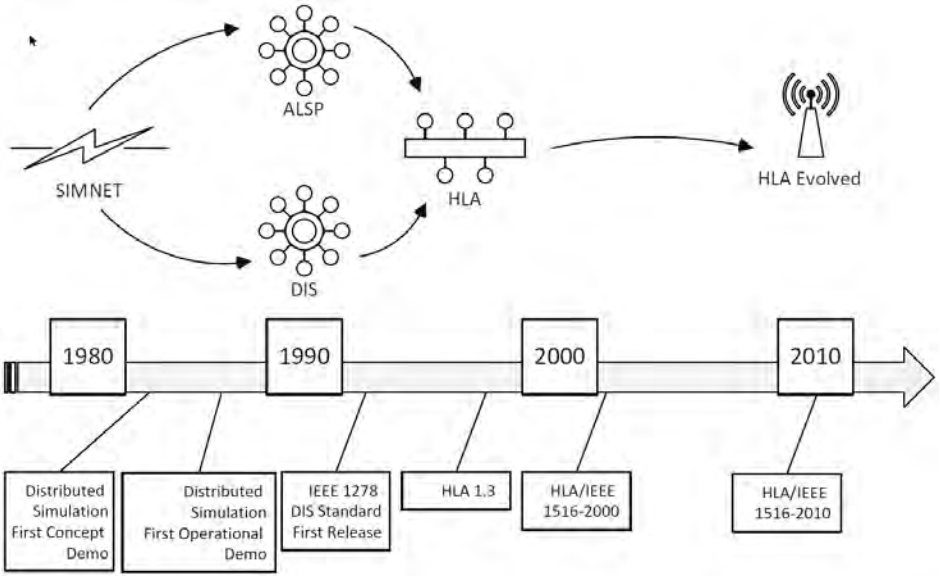

FIGURE 2.19 – Histoire de la simulation distribuée [Oka+16] .

Il fait également référence aux deux autres standards HLA : HLA Object Model Template Specification [IEE10b] et HLA Federate Interface specification [IEE10c].

Le deuxième livret [IEE10b] fournit les spécifications des Object Model qui définissent la structure des informations produites ou demandées (SOM [31]) par une application de simulation (Fédéré) [IEE10a], et la structure (FOM [32]) du référentiel commun de données à l'ensemble des applications de simulations en interaction (Fédération).

Le troisième livret [IEE10c] spécifie les services nécessaires à la connexion des simulations distribuées entre elles. Il définit les opérations, qui doivent être incluses dans une infrastructure logicielle, dénommée RTI [33] (ici, décomposé en "LibRTI et en RTIExec ; figure 2.20), nécessaire à l'exécution des simulations distribuées [IEE10a]. Le RTI est un processus en tant que tel (RTIExec) [Jud98]. Les échanges de données entre Fédéré passent par le mécanisme de communication inter-processus (IPC). Ce mécanisme est à prendre au sens large : par exemple de la mémoire partagée dans un environnement multiprocesseur, une communication réseau (ex : Internet) pour une simulation géographiquement distribuée.

Définition : Le standard HLA définit le terme Fédéré (Federate) comme étant une application capable de se connecter au RTI et de rejoindre une fédération en cours d'exécution. Les Fédéré peuvent être des calculateurs dédiés à la simulation (Simulation), des appli-

31. *Simulation Object Model*
32. *Federation Object Model*
33. *RunTime Infrastructure*

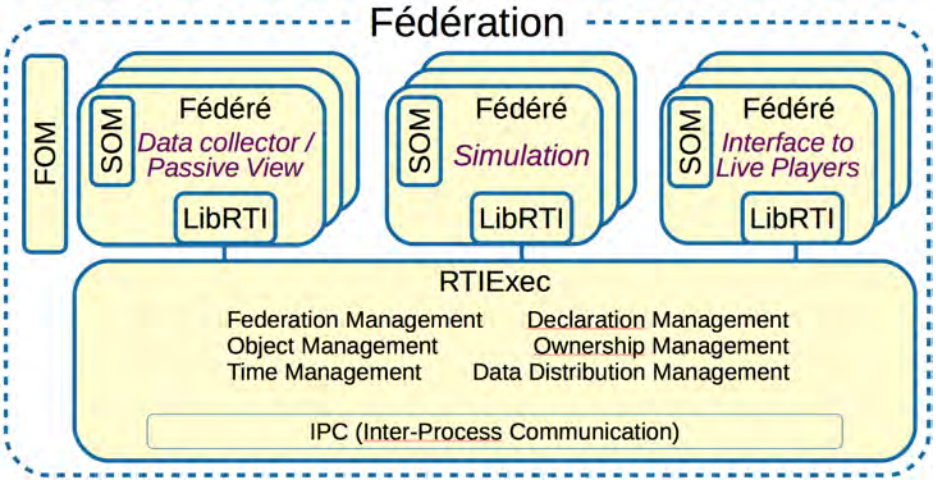

FIGURE 2.20 – Architecture HLA [Jud97] & [DSC12].

cations d'enregistrement et visualisation de paramètres (Data collector / Passive View), tout comme des interfaces en lien direct avec le monde réel (Interface to Live Players) [Jud97]. Si HLA n'impose pas de contraintes particulières quant aux types de simulations pouvant être exécutées par les Fédéré, ce standard impose, toutefois, la nécessité d'inclure les mécanismes permettant aux objets, situés dans les simulations, de communiquer entre eux, via le RTI [Jud97].

Définition : Une Fédération est le nom donné à l'ensemble des Fédéré. À cet ensemble est associé un modèle commun de données FOM, pour atteindre un objectif spécifique de simulation [IEE10a] [Jud97]. Cet objectif peut être, par exemple, de s'assurer qu'une voiture autonome réagisse suffisamment rapidement pour éviter la collision avec un piéton traversant soudainement la route. Dans un tout autre cadre, cela pourrait être la vérification du bon déroulement (simulé) d'un plan d'intervention en cas de catastrophe naturelle : intervention des différents acteurs (préfecture, hôpital, police/gendarmerie, secouriste, etc.), liens de communications fonctionnelles, respect des procédures et vérification de la pertinence des messages échangés, avec la possibilité d'introduire des difficultés : perte liens radio, panne de véhicules.

2.8.4.3 Le temps dans HLA

La gestion du temps est un des atouts majeurs de HLA. Celui-ci ne possède aucune référence à une date partagée par la fédération. Cela veut dire que chaque fédéré doit posséder sa propre référence de temps logique. À charge pour le fédéré de mettre en place une relation entre son temps logique et tout autre référentiel temporel [DSC12]. Pour avancer dans le temps, chaque fédéré fait une demande auprès du RTI qui retourne une

autorisation immédiate ou après un délai, en prenant en compte les horloges logiques des autres fédérés. De cette manière, les principes de causalité des messages entre fédérés sont respectés. Un fédéré ne peut recevoir un message dans son passé.

Plus précisément, HLA supporte 3 mécanismes d'avance dans le temps des fédérés, participant à une fédération. Le premier mécanisme concerne le *temps coordonné*, qui spécifie que l'avance du temps est coordonné avec l'ensemble des autres fédérés de la fédération. Le deuxième mécanisme s'appuie sur les *messages*. L'avance dans le temps d'un fédéré est cadencée par la réception des messages dont une estampille temporelle est accolée à chaque message. Enfin, le troisième mécanisme, dit *optimiste*, considère que chaque fédéré avance à sa propre vitesse, indépendamment les uns des autres (cas du temps réel). Cette approche nécessite un mécanisme de *rollback* pour prendre en compte la réception d'un message avec une estampille temporelle se situant dans le passé du fédéré.

Dans le cas particulier du temps réel, il peut être souhaitable de mettre en place un mécanisme de synchronisation entre l'ensemble des fédérés d'une même fédération.

2.8.4.4 Les avantages et inconvénients

Un des grands avantages de HLA est qu'il intègre, dès sa définition, la notion de simulation distribuée. De plus et de par l'emploi du code exécutable dans les fédérés, le standard HLA assure la protection intellectuelle des modèles de simulation. L'autre grand atout est la présence d'un mécanisme de gestion des temps qui prend en compte les différentes possibilités qu'offre la gestion *coordonnée*, la gestion par les messages pour les événements discrets, comme le temps réel, le tout au sein d'une même fédération. Cela permet d'utiliser des fédérés de natures différentes [DSC12].

Dans une simulation de type distribuée, [DSC12] précise que le manque de performance n'est pas forcement imputable au standard en tant que tel, ici le standard HLA, mais plutôt aux latences introduites par le réseau. À cet effet, il a été montré, en 1998 (projet européen EDISON, projet OTAN [34] « First Wave ») qu'HLA est capable de réaliser des simulations temps réel, même sur des réseaux longue distance [DSC12].

Parmi les inconvénients, le standard HLA est un standard compliqué à utiliser [DSC12]. HLA demande une adaptation spécifique à chaque modèle. C'est une des raison pour laquelle, dans le cadre du projet GENESIS financé par la DGA [35], l'ONERA [36] a développé un cadriciel pour concevoir, développer et générer automatiquement des HLA federate [Bou+05]. À cette difficulté consubstantielle au standard HLA, il apparaît que les RTI

34. *Organisation du Traité de l'Atlantique Nord*
35. *Direction Générale de l'Armement*
36. *Office National d'Études et de Recherches Aérospatiales*

commerciaux (un par fédération, figure 2.20) peuvent être incompatible entre eux, à cause de différences dans la réalisation de l'architecture et protocole de communication entre le RTI et les fédérés. Il en résulte, contrairement à l'objectif initial d'interopérabilité, que des fédérés développés pour un RTI donné peuvent nécessiter une reprise pour fonctionner avec un autre RTI.
Il existe un autre standard de simulation FMI, développé à l'initiative de constructeurs automobiles [FMI14]. À notre connaissance, il n'est pas décrit dans la littérature, les raisons pour lesquelles ce standard a été développé en lieu et place de l'utilisation du standard HLA. Ce standard FMI est décrit dans la section suivante.

2.8.5 Le standard de simulation FMI 2.0

Ce standard a été initié par la société Daimler AG pour favoriser les échanges de modèles de simulation entre les fournisseurs et les équipementiers (OEM [37]) dans le monde automobile. Ce standard FMI, indépendamment des outils, permet de faire de la simulation de modèles dynamiques, suivant deux approches : Model Exchange (FMI 1.0) et Co-simulation (FMI 2.0). Le premier mode FMI 1.0 a été publié en 2010, tandis que le second (FMI 2.0) est rendu public en juillet 2014. Ce standard semble avoir été bien adopté par de nombreux éditeurs de logiciels. Le nombre d'outils disponibles est assez conséquent [FMI19] (section tools). Une description succincte de ces deux standards est présentée dans la section suivante.

2.8.5.1 Quelques fondamentaux de FMI

Pour être exécutés les modèles, au format FMI 1.0, doivent être intégrés à une plateforme de simulation, puisque ces modèles ne contiennent pas leurs propres solveurs d'équations différentielles. De par la nature même de cette approche, ce standard FMI 1.0 ne permet pas de faire de la simulation en entreprise étendue. C'est la raison pour laquelle, ce standard FMI 1.0 ne sera pas plus abordé dans le reste de cette thèse.

À l'inverse, le standard FMI 2.0 impose que les modèles de simulation contiennent leur propre intégrateur d'équations différentielles. Le logiciel contenant le modèle, le solveur et le fichier de description des entrées/sorties de ce même modèle est dénommée FMU [38].

Les bénéfices sont multiples. Le premier est de bien mieux protéger la propriété intellectuelle du fournisseur, puisqu'il n'a pas besoin de dévoiler le type de solveur et les paramètres internes employés, nécessaires à l'exécution de son modèle (figure 2.21-FMU/Slave). Cette

37. *Original Equipment Manufacturer*
38. *Functional Mockup Unit*

protection intellectuelle (IP [39]) permet seulement au fournisseur de visualiser le contenu de son propre modèle, sans pouvoir accéder au contenu des modèles des autres partenaires (figure Fig. 2.22-b). Les figures 2.22-a & figure 2.23-Computer1&2) représentent une autre possibilité offerte par ce standard FMI-2.0 pour réaliser des co-simulations multiples, distribuées et hétérogènes [Nee+14][Gal+15]. Cette co-simulation peut prendre plusieurs formes : l'exécution de FMU simple, tout comme l'exécution d'outils de simulation (figure 2.23-Simulation tool). Ces outils de simulation peuvent être des logiciels spécifiques métiers de partenaires, et dont ils ne souhaitent pas partager le code source et/ou l'exécutable. Cela peut être également du matériel, dont les commandes sont converties au standard FMI. L'algorithme Master contient l'ensemble des moyens de connexion réseau (par exemple CORBA [40], DCOM [41], ou encore le logiciel pour la communication réseau : CosiMate^{TM}, exploité dans le projet MOISE) et les fonctions nécessaires au cadencement de la simulation. Dans cette figure, le FMU sert d'intermédiaire entre la communication réseau et l'outil de simulation.

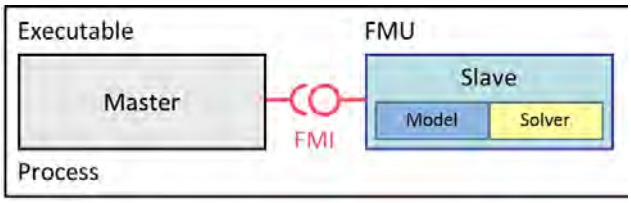

FIGURE 2.21 – Architecture de simulation FMI de base [FMI14].

Il faut noter que le standard FMI-2.0 ne spécifie pas de moyens de communication réseau. La définition de ces moyens est laissée à l'appréciation de l'entreprise réalisant cette co-simulation, ou aux entreprises partenaires dans le cas d'une entreprise étendue.

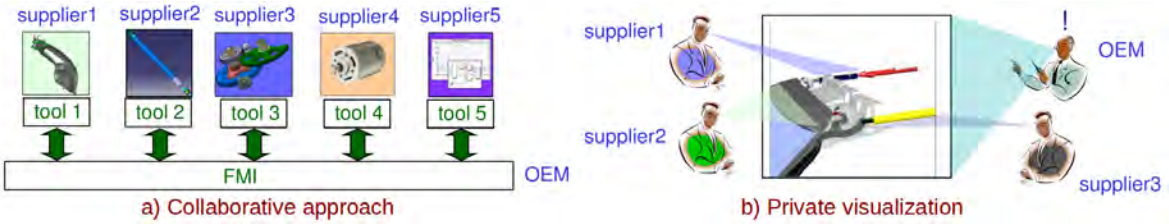

FIGURE 2.22 – Les aspects « collaboratif » et « privé » du standard FMI 2.0 [FMI14].

39. *Intellectual Property*
40. *Common Object Request Broker Architecture*
41. *Distributed Component Object Model*

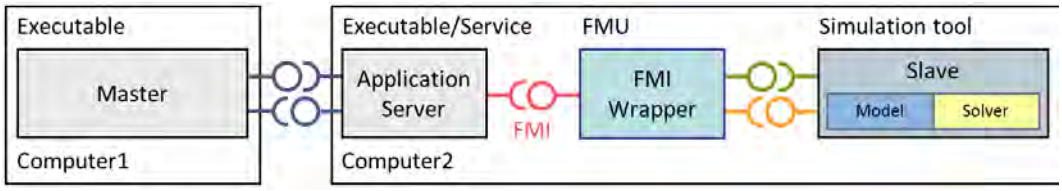

FIGURE 2.23 – Architecture de simulation FMI distribuée [FMI14].

2.8.5.2 La gestion du temps dans FMI

La gestion du temps avec le standard FMI est de type *coordonné* (selon la définition HLA). L'algorithme *master* définit le temps de départ t_{Start} et de fin t_{Stop} du temps simulé, et le pas d'avancement/de communication de la simulation $t_{ci} \rightarrow t_{ci+1}$. À chaque pas de communication, l'algorithme *master* gère les échanges de données entre les différents FMU. À la fin d'un calcul, il lit les données de chaque FMU puis transmet les données récupérées à destination des FMU qui en ont besoin pour le pas de calcul suivant. Cette action est répétée jusqu'à la fin du temps de calcul t_{Stop}.

Si le standard FMI propose un algorithme *master* de base, il semble plus performant de créer un algorithme *master* adapté à la cosimulation, qui prend en compte les différentes branches et boucles de rétroaction du modèle de co-simulation. Il en est de même pour la mise en place des mécanismes de *rollback* [Ler+17].

2.8.5.3 Les avantages et inconvénients

Le grand avantage du standard FMI est sa simplicité (par rapport à HLA, par exemple) et sa large diffusion auprès du monde industriel. C'est une des raisons pour lesquelles ce standard a été retenu par le projet MOISE, d'autant plus que deux membres partenaires du projet ont fourni les outils tant pour la modélisation avec SimulationX^{TM} que pour le moyen de communication réseau entre les FMU CosiMate^{TM}. L'autre intérêt, c'est la présence d'une communauté importante et vivante autour de ce standard.

Parmi les inconvénients, l'API du standard FMI ne fournit pas de mécanismes standards pour gérer les événements et la demande de *rollback*, lorsque l'état du FMU a changé durant un pas de calcul et pour lequel les autres FMU doivent en être informés. Autre difficulté, l'API de ce standard est avant tout une API pour échanger des données scalaires. Il est difficile de passer simplement des paramètres de type vecteur (en prévision pour une version ultérieure de ce standard). Enfin, comme le standard FMI ne précise pas un protocole de communication, pour faire de la co-simulation via le réseau, il faut intégrer ces mécanismes dans l'algorithme *master*, sans oublier de mettre en place les serveurs adéquats sur chaque unité d'exécution.

2.8.6 Le pont entre des standards de simulation

Les sections précédentes présentent deux standards de simulation HLA et FMI, utiles pour l'exécution des modèles de simulation. [Nee+14] crée un pont entre les deux standards FMI et HLA, pour facilité l'emploi du FMI au sein de simulation distribuée de type HLA. Dans le même esprit, [DAC17] étend le standard FMI 2.0 et fournit les outils (indépendant de HLA) pour mettre en œuvre des simulations sur des architectures distribuées et mult-cœurs.

2.9 Cosimulation : Logiciel pour la simulation hétérogène

Dans les sections précédentes, il a été décrit des méthodes ingénierie système et un logiciel dédié à ce type d'ingénierie (Capella) (section 2.7). La section 2.8 a mis en exergue quelques difficultés et quelques standards liés à la simulation. Cette section comble le manque situé entre l'ingénierie système nécessaire à la conception d'un produit et la simulation comme étant une possibilité de vérification/validation des exigences du client.
Cette section aborde les outils permettant de modéliser le comportement attendu des fonctions système : MoI (Modèle d'Intention) pour [Ret+14], ou encore SRM (Simulation Reference Model) pour l'équipe de cosimulation du projet Moise et la possibilité de développer le modèle de calcul maître (« Master ») pour animer une simulation .

Cette section présente trois grandes approches pour réaliser les modèles à destination de la simulation. Modélica fournit un ensemble de bibliothèques multi-métiers. Ces librairies sont la source de nombreux logiciels commerciaux et open source. Le logiciel Ptolemy base son concept de modélisation hétérogène sur la notion de hiérarchie et de modèle de calcul. Le logiciel expérimental ModHel'X reprend le concept de modèle de calcul de Ptolemy et définit les adjonctions de sémantiques pour réaliser les adaptations nécessaires à la simulation de modèles hétérogènes. Enfin, GEMOC Studio développe une autre approche en fournissant des outils pour définir des méta-modèles exécutables et pour synchroniser l'exécution des différents langages ainsi décrits. De par cette technique et après instanciation des méta-modèles exécutables, il devient possible d'explorer par la simulation les différents états d'un système.

2.9.1 Modelica

Modelica est une association à but non lucratif, non gouvernementale, qui regroupe des universitaires comme des industriels. Cette association promeut et développe le langage de modélisation Modelica pour la modélisation, la simulation et la programmation de systèmes et de processus physiques et techniques, dans le domaine des systèmes cyber-physique [Mod18]. Pour ce faire, elle met à disposition, sous forme open source, les spécifications du langage Modelica, un ensemble de librairies pour modéliser, par exemple (non limitatif), les domaines tels que la mécanique, l'électronique, le thermique, ou encore le contrôle. De plus, cette association fournit, maintient et tend à faire connaître, en plus du langage, les standards FMI, SSP (System Structure and Parameterization) et DCP (Distributed Co-simulation Protocol). Des éléments du standard FMI sont donnés dans la section 2.8.5. Le standard SSP, quant à lui, décrit la manière dont les composants d'un modèle peuvent être assemblés et connectés pour former des composants composites. Ces composants peuvent être présents sur plusieurs unités d'exécution et sont compatibles au standard FMI. Le standard DCP est un protocole de communication qui fournit la brique manquante pour que les modèles au format FMI puissent communiquer entre eux, via le réseau. Plus exactement, ce protocole offre des possibilités bien plus vastes. Il est conçu pour intégrer des modèles et des systèmes temps réel dans des environnements de simulation. Ce protocole s'appuie sur les couches de communication UDP (User Datagram Protocol), TCP (Transmission Control Protocol) ou encore sur le bus CAN (Controller Area Network). De par sa conception, ce protocole DCP est compatible au standard FMI.

Autre point, le langage Modelica permet de modéliser des systèmes hybrides discret-continu, au sens où le discret peut être amené à piloter le continu, toujours dans un déroulement temporel continu de la simulation [Pto14]. Le rebondissement d'une balle sur le sol en est l'exemple classique. La partie discrète est constituée par équation en charge de la détection de la balle touchant le sol. Lorsque le sol est atteint, la partie continue, c'est à dire l'équation gérant le déplacement de la balle est modifiée pour signifier le rebond [Rei17] .

Cet écosystème agrège un grand nombre d'éditeurs logiciels commerciaux ou à distribution open source tel que OpenModelica.

2.9.2 Ptolemy

Ptolemy est un cadriciel développé par l'université de Berkeley pour concevoir, modéliser et réaliser des simulations hétérogènes [Pto14]. Ce cadriciel s'appuie sur cinq concepts fondamentaux : les concepts d'*Acteur*, de *MoC* (Model of Computation), de *Directeur* au sens du cadencement des activités, de hiérarchie et de simulation de modèles hétérogènes.

L'*Acteur* (figure 2.25) est un composant logiciel qui peut s'exécuter concurremment aux autres et communiquer par des messages, via ses ports d'interconnexion. Il contient la sémantique comportementale du modèle. Le MoC est un ensemble de règles qui peuvent être décomposées en trois catégories. La première définit les constituants d'un composant, c'est-à-dire d'un *Acteur*. Le second ensemble de règles précise, spécifie les mécanismes d'exécution et de concurrence, tandis que le troisième ensemble est centré sur les principes qui gouvernent la communication entre les composants. Le *Directeur* implémente le MoC. Ptolemy propose plusieurs types de *Directeurs*, et donc plusieurs types de cadencement de la simulation, dont une représentation taxonomique en est donnée par la figure 2.24. Pour illustrer en partie cette diversité, il est possible de citer le *Process Networks* (PN)

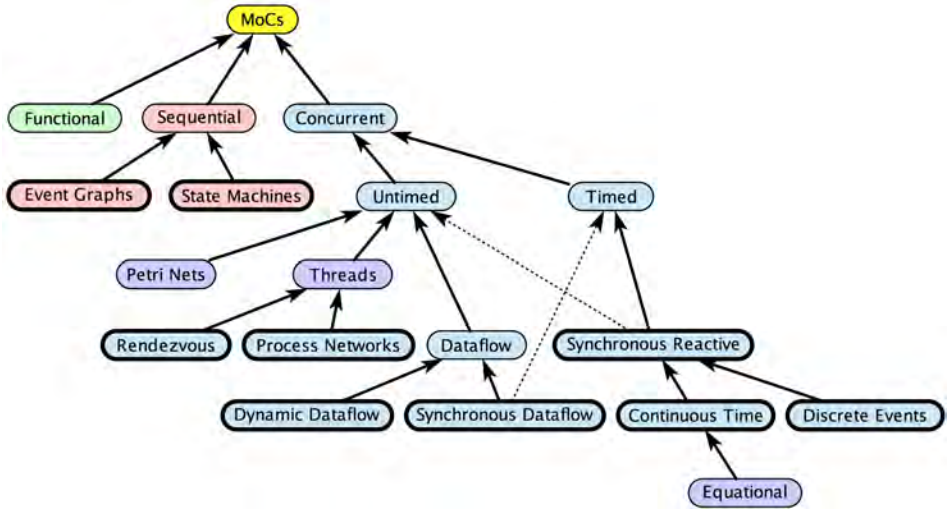

FIGURE 2.24 – Taxonomie des *Model of Computation* [Pto14].

qui est bien adapté pour décrire les processus concurrents qui communiquent entre eux par messages asynchrones. L'émetteur n'attend pas d'accusé réception de la part du récepteur. Ou encore le *Directeur* événement discret (DE : Discrete Event) permet aux *Acteurs* de communiquer par événements horodatés pour assurer la causalité du traitement des messages. Le *Directeur* flux de données (DF : Data Flow) est bien adapté aux séquences de données tels les flux audio ou vidéo : l'*Acteur* réagit à la disponibilité de données sur ses entrées. Avec les flux de données synchrones (Synchronous Data Flow : SDF), la quantité de données à lire en entrée est préalablement fixée . Avec les flux de données dynamiques (Dynamic Data Flow : DDF), la consommation des données dépend du schéma de données présent en entrée.

L'autre atout important de Ptolemy est la décomposition hiérarchique des modèles, dont le principe est représenté par la figure 2.25. Dans cette figure l'*Acteur* A est considéré comme un *Acteur composite opaque*, car son comportement est raffiné, dans un niveau hiérarchique inférieur, en un *Acteur* D géré par son propre *Directeur*. L'*Acteur* C, quant à

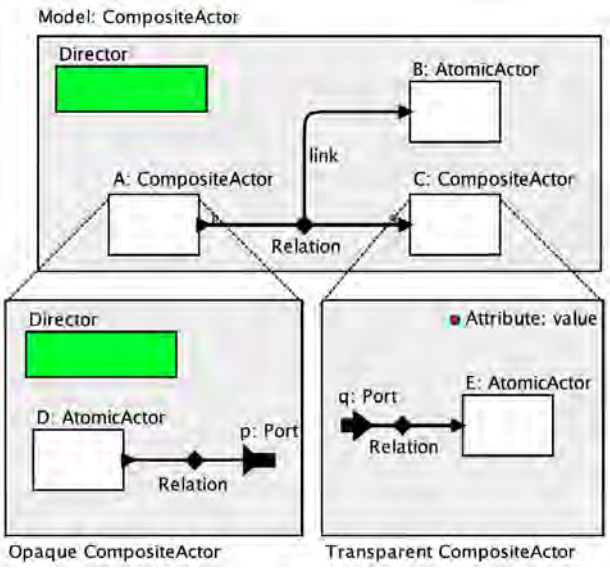

FIGURE 2.25 – L'aspect hiérarchique de Ptolemy [Pto14].

lui, est un *Acteur* dit transparent dans le sens où il est décomposé en un *Acteur* E gouverné par le directeur du niveau supérieur. L'intérêt de cette approche hiérarchique est de pouvoir faire des simulations hétérogènes dans le sens où les directeurs, qui possèdent leur propre sémantique forte, peuvent être de nature différente, à chaque niveau de la hiérarchie.

Le modèle hiérarchique s'applique également à la gestion du temps. C'est le point clef qui permet d'assurer l'inter-opérabilité entre différents domaines de MoC (ex : modèles à temps continu ou encore machine à états finis). Chaque modèle dans la hiérarchie prend en compte suivant son besoin, le temps qui lui est fournit par le *Directeur* de plus haut niveau. Pour ce faire, Ptolemy fournit une notion cohérente du temps (logique) [Pto14] dite *superdense time*. C'est une paire de valeur $(t,\ n)$ dénommé « estampille temporelle » où t représente le modèle de temps logique et n l'index. Le modèle de temps t représente l'heure à laquelle un événement se produit, tandis que l'index n représente le séquencement des événements qui se produisent pour cette même heure. Ptolemy définit la notion de simultanéité, au sens large, pour deux estampilles temporelle $(t1,\ n1)$ et $(t2,\ n2)$ si $t1 = t2$, même si *n1 est différent de n2*. La notion de simultanéité forte nécessite que $t1 = t2$ et $n1 = n2$. Pour résoudre les problèmes dus à la sémantique des nombres qui peuvent être différentes suivant les environnements de développement, Ptolemy propose un pas de résolution temporelle, comme étant une constante global à l'ensemble des modèles. La représentation du modèle de temps prend alors la forme : $t = mr$, ou m est un entier long, et la résolution r est un nombre flottant en double précision, partagé par l'ensemble des *Directeurs*. De cette manière, cette résolution temporelle se propage à travers toute la hiérarchie des modèles. Par défaut, Ptolemy pose la résolution temporelle à 10E-10, soit 1/10 de nano-seconde [Pto14].

2.9.3 ModHelX

Dans un modèle hétérogène, c'est-à-dire dans un modèle comportant des parties de modèle ayant des formalismes différents, par exemple, de type : DE : Discrete Event, FSM : Finite State Machine ou encore SDF : Synchronous Data Flow et autres [Sup14], il est souvent nécessaire d'ajouter des éléments d'adaptation entre ces formalismes pour rendre efficace leurs interrelations. En d'autres termes, permettre d'adapter les sémantiques des données, de contrôle et temporelles. C'est notamment le cas dans les environnements de simulation tels que Ptolemy, ou encore Simulink. Ces adjonctions sont dénotées *adaptation diffuse*, dans le sens où elles peuvent être réparties dans l'ensemble du modèle. Elles peuvent être difficiles à spécifier et elles modifient de facto le modèle [Mey+13]. La figure 2.26 en donne un exemple, en se basant sur le formalisme de modélisation de Ptolemy. Cet

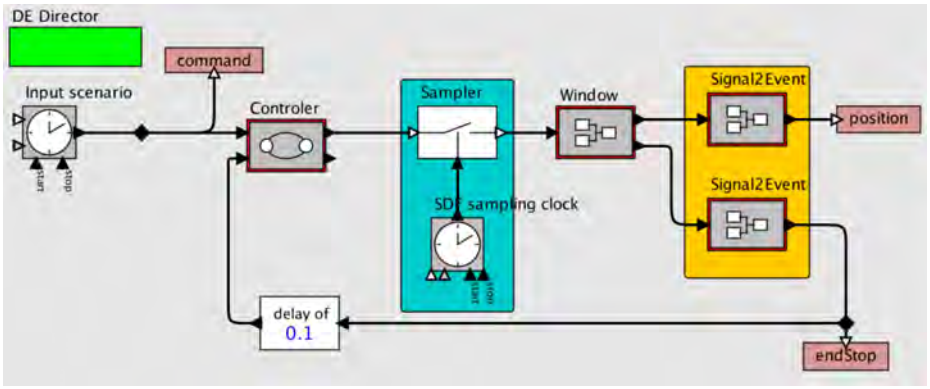

FIGURE 2.26 – Adaptation diffuse [Mey+13].

exemple [Mey+13] [Bou+11] décrit les mécanismes mis en œuvre pour monter/descendre une fenêtre de voiture. La chaîne de commandement de base est constituée par le bouton-poussoir de commande (modélisé ici par le composant *Input scenario*), le contrôleur d'état (*Controler*) qui gère cette commande et la prise en compte de la position de la fenêtre lorsque celle-ci est arrivée en point haut ou bas, et enfin, le mécanisme de montée/descente de la fenêtre (*Window*). Comme vu dans la sous-section précédente (sous-section 2.9.2) concernant Ptolemy, le MoC gère l'exécution des modèles de son périmètre. Dans le périmètre de la figure 2.26, il y a trois formalismes : DE (Discret Event) (*Input scenario*), TFSM (Timed Finite State Machine) (*Controler*), SDF (Synchronous Data Flow) (Window). Les deux premiers sont efficacement gérés par le MoC *DE Director* puisque leur fonctionnement est basé sur des événements. Le composant *Window*, basé sur le formalisme du flux de données synchrones SDF [42], ne peut être correctement exécuté par un MoC de type *DE*, puisque l'entrée du composant *Window* attend un flux de données, alors que cette même entrée reçoit des événements de la part du *Controler*. Il en est de même pour

42. *Synchronous Data Flow*

la sortie de ce composant *Window* qui fournit un flux de données, lorsque le MoC attend un événement. C'est la raison pour laquelle, il est nécessaire d'introduire une adaptation dénommée *Sampler* (avec son horloge : SDF Sampling clock) entre le *Controler* et le composant *Windows*. De cette manière, le composant *Window* reçoit en permanence un flux de données correspondant à l'état du *Controler*. Cette adaptation est également nécessaire en sortie de ce composant *Window*, pour transformer le flux de données en événement, via l'adjonction des deux composants de transformation : *Signal2Event*.

L'équipe de ModHel'X propose une approche différente, via un environnement expérimental de développement, pour concevoir, mettre au point, tester et simuler des modèles de simulation, dont les parties peuvent combiner des formalismes différents. Le but est d'éviter les *adaptations diffuses* et donc toutes modifications du modèle initial. Pour ce faire, l'équipe ModHel'X s'inspire du MoC de Ptolemy pour la prise en compte des divers formalismes, définit un méta-modèle indépendant des MoC pour décrire la structure d'un modèle, et conçoit un moteur d'exécution générique pour les MoC inspirés.

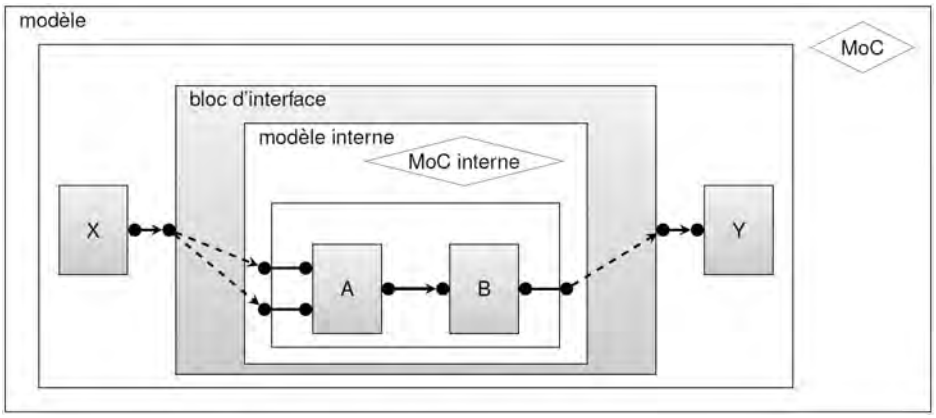

FIGURE 2.27 – Interface d'adaptation [JHB14].

Le méta-modèle définit les concepts de *Bloc*, *Pin* et *Relation*, (voir figure 2.27 pour les concepts instanciés) [JHB14]. Le *Bloc* est l'unité élémentaire de comportement (zone rectangulaire). Il s'agit d'une boîte « noire » dont l'interface est spécifiée. La patte *Pin* (point noir) est le mécanisme d'échange d'information entre blocs. L'ensemble des *Pin* d'un *Bloc* forme l'interface de ce *Bloc*. Le concept *Relation* connecte les *Pin* entre elles (trait noir entre deux points). Sur la base de ce méta-modèle, ModHel'X définit ce qu'est un modèle : un ensemble structuré de *Bloc* associé à un MoC [Bou+11]. A cette définition initiale, l'équipe de ModHel'X ajoute que l'hétérogénéité dans les modèles multiformalismes est obtenue en structurant hiérarchiquement les modèles. La figure 2.27 traduit bien cette hétérogénéité par la présence d'un modèle (*modèle*), composé d'un *Bloc* et de son MoC, englobant un autre *Bloc* dénommé *modèle interne* qui contient lui-même un *Bloc* ayant une structure composée des *Bloc* A et B.

Toutefois, il est à noter que l'idée originale dans la figure 2.27 est située entre les modèles *modèle* et *modèle interne*. C'est le *Bloc* dénommé *bloc d'interface*. Ce *Bloc* permet pleinement de mettre en œuvre l'hétérogénéité par décomposition hiérarchique en ayant recours à l'adaptation des sémantiques entre les deux MoC englobant et interne. En se basant sur la figure 2.26 pour illustrer, trois sémantiques sont à prendre en compte [JHB14] :
- la première adaptation consiste à transformer la donnée de type TFSM [43] issue du *Controler* en une donnée de type flux à destination du composant *Window*, tout en adaptant la sémantique du nombre (entier, réel, flottant, etc.),
- L'autre sémantique concerne le contrôle des instants d'observations. En se basant sur l'exemple de la figure 2.26, le *Controler* n'est observé que lorsque qu'il transmet son changement d'état, alors que le composant *Window* observe à chaque instant son entrée, suivant une période d'échantillonnage donnée.
- Enfin, la dernière adaptation sémantique concerne la notion de temps. En effet, les modèles et sous-modèles peuvent avoir des échelles de temps différents [JHB14].

Pour compléter ces informations, l'environnement de développement expérimental ModHel'X et la documentation associée sont disponibles sur le site de Centrale Supélec à l'adresse : https ://wdi.supelec.fr/software/ModHelX/

Ces préoccupations quant à l'adaptation des modèles aux différents types de formalisme se retrouvent de manière encore plus développées dans le logiciel GEMOC [44] Studio, dont une description est donnée dans la sous-section suivante.

2.9.4 GEMOC Studio

Les ingénieurs modélisent un système suivant leur point de vue métier, en créant des DSML [45] (Langage de modélisation spécifique à un domaine). Une de leurs préoccupations est de pouvoir simuler, au plus tôt, le système suivant leur modèle métier, ou encore avec l'ensemble des points de vue métier. Parmi les différentes possibilités de simulation, l'une des manières de faire est celle proposée dans cette thèse, par la création d'une architecture de simulation, et par le développement d'exécutable de simulation spécifique à chaque fonctionnalité identifiée. Pour traduire les modèles d'intention (MoI) ou SRM [46] de l'équipe MOISE, une autre approche, franchement différente, est proposée par le groupe de recherche *Gemoc* [Bou+16]. Cette approche est basée sur l'exécution coordonnée des langages de modélisation. Elle consiste à rendre directement exécutable les langages de modélisation spécifiques à un domaine (DSML) et à assurer leur exécution coordonnée. Comme

43. *Timed Finite State Machine*
44. *The GEMOC Initiative On the Globalization of Modeling Languages*
45. *Domain Specific Modeling Language*
46. *Simulation Reference Model*

pour les DSL, ce langage contient une syntaxe abstraite et concrète. A ces syntaxes, l'approche *Gemoc* propose d'ajouter une sémantique opérationnelle à chaque concept défini dans les langages de modélisation, puis d'expliciter la coordination entre les différents langages ainsi définis. Les résultats de recherche sont, en autres, regroupés dans l'outil *Gemoc Studio*, qui est disponible en open source pour l'ensemble de la communauté [GEM19a], à travers la fondation Eclipse. Une vue d'ensemble de cet outil est donnée par la figure 2.28.

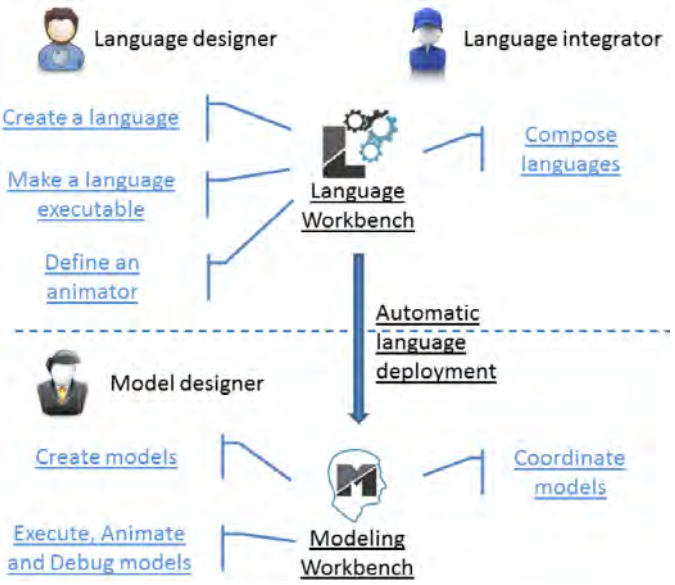

FIGURE 2.28 – Vue d'ensemble de l'atelier Gemoc-Studio [GEM19a].

Ce cadriciel *Gemoc Studio* s'appuie sur la technologie *Eclipse*, pour concevoir l'exécution coordonnée de différents modèles métiers (DSML). Ce cadriciel est décomposé en trois parties, dont deux sont représentés dans la figure 2.28. La première partie (non représentée) contient sous forme de méta-langages des services communs utiles pour l'exécution de ces DSML. On y retrouve le méta-langage pour la coordination d'exécution des modèles hétérogènes (BCOoL : Behavioral COordination Operator Language [Var+15]), le méta-langage pour l'exécution concurrente des DSL (MoCCML : Operational Semantics of the Model of Concurrency and Communication Language [Dea+14]), le méta-langage pour rendre les DSML exécutables suivant la technologie fUML (xMOF : Executable DSMLs Based on fUML [TUW13] [May+13]) et sur différents logiciels complémentaires pour gérer l'animation des modèles, basés sur Sirius, la gestion des traces d'exécution, des débogueurs séquentiel et concurrentiel.

La figure 2.28 représente les deux étapes majeures pour réaliser des modèles métiers exécutables. Cela commence par la définition des langages de modélisation spécifiques aux domaines visés ("Create a language"), puis par la définition de la sémantique opérationnelle ("Make a language executable"), la définition de la sémantique concrète d'animation des

modèles ("Define animator") et enfin par la coordination de l'exécution de ces différents langages précédemment définis ("Compose languages").

Lorsque cette composition de langages est réalisée, le code est automatiquement généré pour permettre aux concepteurs de créer leurs modèles ("Create Models"), d'animer leurs modèles ("Execute, Animate and Debug models") et la coordination de leur exécution ("Coordinate Models") en utilisant les opérateurs précédemment définis, au niveau langage, avec *BCOoL*.

Même si cette approche *Gemoc* propose un cadre théorique important pour réaliser des simulations, avec des langages hétérogènes, cette approche n'est pas exploitée dans la suite de cette thèse, qui repose sur les moyens actuellement utilisés pour le développement de simulateurs dans le monde industriel. Notre proposition se plaçant à un niveau méthodologique, elle reste compatible avec une utilisation potentielle de *Gemoc* et des autres technologies évoquées.

2.10 Une articulation entre l'IS et les activités de simulation

L'ingénierie système est une démarche de moyens et une méthodologique interdisciplinaire, pour réaliser des systèmes performants avec succès [INC15]. (cf section 2.6). Actuellement, cette démarche s'appuie sur des outils d'ingénierie système basés modèles (section 2.7.3), pour assurer la cohérence entre les différentes vues du système, pour aider à la prise de décision, à la vérification et à la validation du produit désiré. Il n'en reste pas moins que les modèles du système produits, par cette démarche, peuvent être qualifiés de contemplatifs, vis-à-vis de leurs dynamiques.

La démarche de simulation, quant à elle, tend à donner « vie » aux modèles contemplatifs de l'IS. En ce sens et dans le cas présent (de cette thèse), cette démarche de simulation peut être considérée comme orthogonale au concept d'IS (figure 2.29).

Cette orthogonalité ouvre un premier espace plan dans lequel les modèles contemplatifs d'IS et la démarche de simulation peuvent être liés pour créer, puis exécuter, les modèles dynamiques issus de cette IS. La troisième dimension de cette figure 2.29 représente la profondeur de champ de raffinement des modèles des fonctions d'IS pouvant tendre vers un jumeau numérique, du produit désiré. Par jumeau numérique, il est possible de comprendre que le modèle numérique exécutable est réalisé, avec la finesse nécessaire, pour pouvoir y être interrogé et fournir des réponses conformes à la réalité, dans les intervalles de tolérances spécifiés.

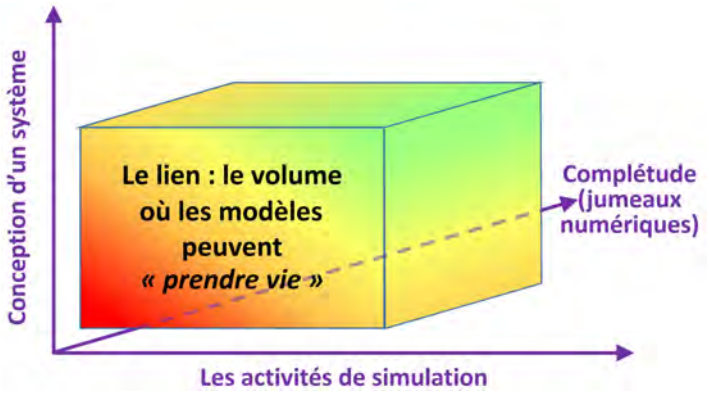

FIGURE 2.29 – Articulation possible entre l'IS (basée modèles) et la simulation.

Le but de cette thèse, en regard de ce qui est exposé ci-dessus, est de créer le lien méthodologique, basé MBSE, entre la démarche d'ingénierie système et la démarche de simulation, pour organiser, structurer la création, le raffinement des modèles exécutables, et assurer la traçabilité entre les modèles d'IS et de simulation, le tout pouvant tendre vers la création de jumeaux numérique. Cette méthodologie d'organisation et de structuration devient prégnante dans le cadre de l'entreprise étendue, où chaque partenaire développe ses modèles et réalise ses simulations au sein de son entreprise.

2.11 Conclusion

Ce chapitre a retracé une histoire, partielle, des évolutions sur la manière de produire (production artisanale, production de masse), sur le déplacement d'une production verticale vers une production de sous-traitance (de capacité, de spécialité, et parfois de substitution) puis de systémier, sur la modélisation/simulation (déjà pratiqué chez les Grecs), et sur les notions, de compliqué et de complexe, appliquées aux réalisations faites par l'Homme. Cette mise en perspective, « savoir d'où l'on vient » a cherché à mettre en évidence que la proposition de thèse s'inscrit assurément dans une préoccupation, aujourd'hui représentée par les industriels, pour réduire les coûts de production en construisant cette fameuse usine 4.0. Le projet MOISE, décrit dans le chapitre précédent sur le contexte industriel de cette thèse, en est l'expression même : intérêts sur l'ingénierie système, la modélisation, la simulation, et le partage des données entre les sous-traitants, systémiers et le donneur d'ordres, pour concevoir et construire un produit de type industriel.

Un des points communs à l'ensemble de ces sections de ce chapitre et quelles que soient les époques, c'est la recherche de la réduction des dépenses pour réaliser un produit. Cela se traduisait par le passé dans des réductions de dépenses musculaires, énergétiques, et de coûts financiers de nos jours.

Dans son article sur les concepts de *l'entreprise en réseau*, [Ror05] propose une vision structurante des différentes typologies d'entreprises en réseau, ici dénommée *l'entreprise étendue*, en relation avec des critères issus du terrain. Elle fait la distinction entre les aspects juridiques et organisationnels de l'entreprise en réseau. Deux typologies sont retenues dans cette thèse, la première reflétant le tissu industriel local (réseau type fédéré : donneur d'ordres à sous-traitants), la deuxième, à titre informatif, expression de l'organisation de l'IRT Antoine de Saint Exupéry : le réseau nucléique.

Concevoir et réaliser un projet peut être compliqué, voire complexe. Cela nécessite d'employer des méthodes, ou plus exactement de mobiliser les principes d'ingénierie système, tant pour s'assurer de la pertinence de la réponse du produit conçu et réalisé en termes de qualité, vis-à-vis de la demande du client, que pour ses coûts et délais de réalisation. Un bref historique de cette ingénierie système a permis de sélectionner le standard d'ingénierie système EIA-632, comme semblant être le plus approprié au contexte de cette thèse. Ce standard embrasse le recueil des exigences jusqu'à la livraison du produit.

Toutefois, avant d'aborder l'ingénierie système dirigée par les modèles, il a semblé utile de faire un point sur l'ingénierie dirigée par les modèles. Il en ressort de cet état de l'art que la mise en œuvre des modèles, basée sur des langages, permet de passer du monde des modèles contemplatifs à des modèles opérationnels, c'est-à-dire, de pouvoir poser des règles de bonne construction d'un modèle, d'en vérifier les incohérences, d'exploiter ces modèles en automatisant les tâches, pour générer du code et toutes autres activités nécessaires. L'ingénierie système dirigée par les modèles bénéficie à la fois de la partie structurante de l'ingénierie système et de l'aspect opérationnel de l'ingénierie dirigée par les modèles.

Dans cet état de l'art, une attention a été portée sur différents types de simulation, sur la gestion du temps, sur le logiciel d'ingénierie système *Capella*, sur les standards de simulation FMI et HLA, sur les logiciels permettant d'exprimer le comportement attendu attaché aux fonctions système, tels que *Modelica*, *Ptolemy*, *ModHel'X* et *Gemoc*. Le logiciel scientifique *Gemoc* (non retenu dans le projet MOISE, car trop éloigné de la pratique industrielle actuelle) est dédié à l'exploration, par la simulation, des interactions et possibles coordinations entre différents DSL/DSML. En accompagnement du logiciel *Capella*, l'emploi du logiciel *Gemoc* permettrait, en amont, de définir, vérifier et valider des méthodologies d'interaction et de coordination entre différents secteurs métiers (par exemple, entre l'IS et la sûreté de fonctionnement) pouvant être mise en œuvre sous la forme de point de vue, au sein du logiciel *Capella*, tout en y apportant des possibilités de simulation. À ces interactions métiers, il faut ajouter la possibilité de vérifier et valider le comportement des modèles ayant des sémantiques hétérogènes.

Suite à cet état de l'art, et à notre connaissance, il n'existe pas de méthode de références commune offrant une vision globale et structurée permettant de définir les interfaces et les comportements des modèles, tout en prenant en compte la création et l'assignation des

modèles sur la plateforme de co-simulation.

Pour palier à ce problème et à l'instar du développement des produits, nous proposons d'utiliser le MBSE pour définir une méthode pour le développement de modèles de simulation et leur exécution. L'autre point contributif important est également le développement d'une plateforme de cosimulation de systèmes en entreprise étendue, en s'appuyant sur une méthodologie MBSE. À ces deux points : méthodes et plateforme, il est nécessaire d'ajouter de nouveaux acteurs ou de redéfinir le rôle de certains d'entre eux, tant pour la partie méthodologie que pour la conception/réalisation de la plateforme de cosimilation.

L'ensemble de ces propositions contributives sont plus amplement décrites et expliquées dans le chapitre suivant (chapitre 3).

Chapitre 3

Propositions - Contributions

Sommaire

3.1 Introduction

Les objectifs de cette proposition de méthode, pour le développement MBSE de simulateur, à partir des modèles d'architecture du produit sont multiples : améliorer la productivité (maîtrise des coûts, anticipation des changements, ...), la compétitivité (produit innovant, qualité des produits et services, ...) et donner à l'architecte système du produit, la possibilité d'acquérir de la confiance dans l'architecture du produit qu'il définit.

Pour remplir ces objectifs, cette proposition se base sur la définition d'une méthode de développement de simulateurs, sur la spécification, la caractérisation MBSE d'une architecture générale d'un système de simulation en entreprise étendue, et sur la création ou la redéfinition du rôle d'acteurs intervenant dans la réalisation des simulateurs, comme dans la réalisation et l'utilisation de cette architecture générale de simulation.

Cette méthode intègre également les différents corps métiers invoqués dans la conception du produit (sécurité, sûreté, mécanique, thermique, etc.), pour la réalisation des simulateurs.

Pour que le développement des simulateurs puisse se réaliser avec un minimum de difficultés, il apparaît nécessaire de devoir faire une séparation claire des différentes responsabilités mobilisées, afin que « nul ne soit juge et partie » (figures 3.3 & 3.4). En premier lieu, l'architecte système (SyA [1]) conçoit le produit. Il est à l'origine des demandes de simulation, pour évaluer son architecture et s'assurer de son adéquation avec les exigences exprimées par le client. Ces demandes seront faites auprès de l'architecte système de la simulation (SiA [2]), qui définit l'architecture de simulation et précise l'emplacement pour l'exécution des différents modèles de simulation au sein de l'entreprise étendue. Les développeurs (SMD [3]) sont chargés de la réalisation des modèles de simulation, puis l'exécution de la simulation est confiée aux responsables de simulation (SEM [4]), de chaque entreprise partenaire. Les responsables des plateformes de simulation (IPM [5]), au sein de chaque entreprise, créent puis s'assurent de la disponibilité des moyens de simulation.

À cette clarification sur le rôle des acteurs, il est utile de préciser les concepts entrant dans cette architecture générale de simulation. Quatre grands concepts sont à préciser : le *modèle de simulation*, la *plateforme de simulation*, la *plateforme de cosimulation* et le *simulateur*. C'est important d'y porter attention, car la réalisation d'un modèle de simulation, la réalisation d'une architecture de simulation peuvent être des projets en eux-mêmes, avec leurs contraintes de coûts, de performances et de délai de réalisation. Cette clarification

1. *System Architect*
2. *Simulation Architect*
3. *Simulation Model Developper*
4. *Simulation Execution Manager*
5. *Infrastructure Project Manager*

est d'autant plus prégnante lorsqu'elle s'exerce dans le cadre d'une entreprise étendue : la plateforme de cosimulation est alors répartie chez les différents partenaires du projet (Fig 3.10).

En première approche et pour fixer rapidement les idées, le *modèle de simulation* regroupe l'ensemble des fonctions modélisées exécutables, pour un objectif de simulation donné.

Au plus haut niveau de l'entreprise étendue, la *plateforme de cosimulation* est constituée par l'agrégation des plateformes de simulation, aussi complexe soit-elle, de chaque partenaire de cette entreprise étendue, qui produit des modèles de simulation pour les parties du système dont il a la charge. Ici, le terme cosimulation signifie la simulation de différents points de vue métiers, puisque chaque partenaire possède une spécificité métier. Une *plateforme de simulation* est l'ensemble des moyens matériels et logiciels qui permettent, à minima, d'exécuter des fonctions modélisées exécutables. Le terme *simulateur* signifie l'exécution des modèles de simulation sur la plateforme de cosimulation. Il y a donc autant de simulateurs que d'objectifs de simulation, alors qu'il n'y a qu'une seule plateforme de cosimulation, pour une entreprise étendue donnée. Pour ne pas être restrictif et suivant la diversité des points de vue métiers exercés par une entreprise, celle-ci a bien évidemment la possibilité d'agréger ses propres plateformes de simulation, pour n'en faire qu'une, ou encore, d'offrir plusieurs plateformes de simulation à l'entreprise étendue.

La section 3.2 aborde spécifiquement la méthode MBSE pour le développement de simulateurs à partir des modèles du produit, tandis que la section suivante 3.3 présente l'étude MBSE de la plateforme de cosimulation qui précise les concepts et la mise en œuvre de cette plateforme de cosimulation, nécessaire à l'exécution des simulations. Un exemple d'instanciation de l'architecture générale de simulation et de la méthode pour créer des simulateurs est donné dans la section 3.4. La section 3.6 porte sur le retour d'expérience de l'équipe MOISE pour la simulation. En complément et plus particulièrement à destination des industriels, la section 3.7 liste un ensemble de propositions de réflexions pouvant être prises en compte, lors de l'élaboration d'une plateforme de cosimulation.

3.2 Méthode de création des modèles de simulation

3.2.1 Introduction

La méthode proposée se veut générique, pour la création des modèles de simulation. Elle peut s'appliquer aux études de phases amont pour simuler des expressions de besoins, comme à la conception/définition du produit pour évaluer une architecture, ou véri-

fier/valider la satisfaction des exigences de besoin du client. Cette méthode prend aisément en compte les différentes approches MBSE pour l'ingénierie système, tout comme les outils dédiés à l'ingénierie de la simulation. De par sa conception générique, cette méthode a pour vocation à s'intégrer dans le processus de développement du produit, quel qu'il soit. De plus, pour réaliser ces simulations, cette méthode prend également en compte l'architecture et les unités d'exécution mises à disposition de l'entreprise étendue.

La conception d'un produit est une activité complexe. Cela commence par le recueil des exigences du client (Phase 1, figure 3.1). L'ingénieur système reprend ces exigences pour définir l'architecture du produit (Phase 2, figure 3.1) et fournir les spécifications adéquates aux personnes en charge de la réalisation des équipements introduits dans l'architecture et dont l'intégration constituera le produit désiré (Phase 3, figure 3.1).

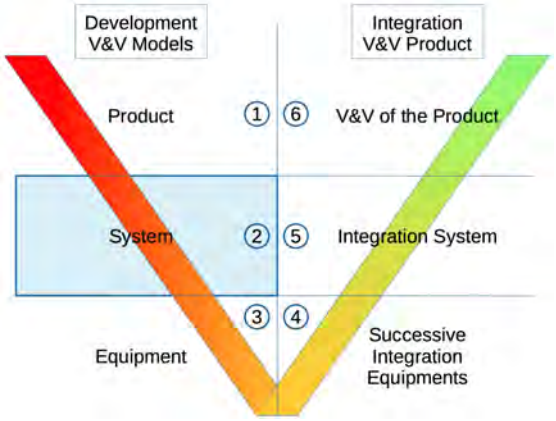

FIGURE 3.1 – Cycle en V pour la réalisation d'un produit.

Or, il est habituel, en parallèle de la conception de l'architecture du produit (Phase 2, figure 3.1), de créer un modèle de simulation. Ce modèle de simulation, permet l'exploration d'une conception, l'évaluation de la conformité de cette conception à l'aune des exigences du besoin, ou encore et in fine, l'estimation du comportement de ce produit.

Or, souvent dans la pratique industrielle actuelle (figure 3.2 : Current State of Practice), la création du modèle de simulation s'appuie directement sur les exigences représentées dans des documents, sous formes textuelles ou de graphiques contemplatifs. Une telle démarche peut occasionner des écarts entre les modèles du produit et les modèles de simulation. Ces divergences peuvent être dues à une architecture et à une interprétation différente des exigences, à un manque de synchronisation lors de l'évolution de la conception du produit, non reportée dans le modèle de simulation.

Pour pallier ces difficultés et minimiser autant que possible les écarts potentiels entre le modèle du produit et de simulation, notre proposition prend appui sur une démarche MBSE et sur l'utilisation d'un unique modèle du produit pour réaliser le modèle de simulation

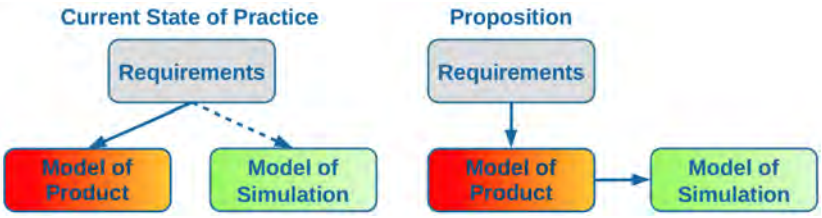

FIGURE 3.2 – État actuel et proposition.

(figure 3.2 : Proposition). Cette démarche MBSE, tant pour la définition de l'architecture du produit que pour la définition de l'architecture de la simulation, permet d'exploiter les modèles pour formaliser, rationaliser, organiser le travail nécessaire à la modélisation de l'architecture de simulation et à son évolution (Proposition, figure 3.2), dans la perspective des objectifs initiaux. La section suivante aborde les principes fondateurs de cette méthode.

3.2.2 Les principes fondateurs de la méthode

Le premier principe fondateur de la méthode (figure 3.3) est de séparer clairement les préoccupations de l'ingénieur architecte du produit (SyA), des personnes en charge de concevoir l'architecture de la simulation (SiA), de réaliser les modèles de simulation (SMD) puis d'exécuter la simulation (SEM). Derrière cette séparation claire des responsabilités, il y a l'idée, qui semble importante dans la pratique du quotidien : « Nul ne peut être juge et partie ».

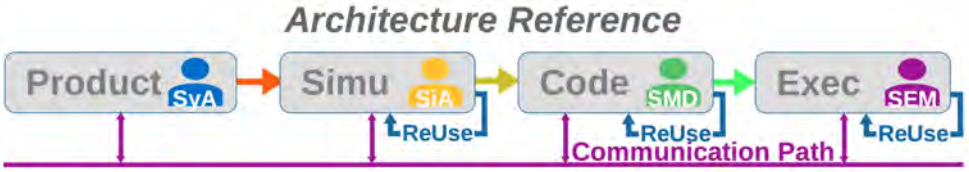

FIGURE 3.3 – Architecture de référence de la méthode proposée.

La deuxième idée fondatrice est de concevoir les simulateurs comme des projets. Or, le nombre de simulateurs à créer est généralement important, tout en ayant des objectifs de simulation proches. Cette multitude et proximité peuvent amener à des pertes de cohérence entre ces simulateurs. L'utilisation des transformations de modèles et l'exploitation de leurs propriétés associées peuvent permettre de rationaliser l'ensemble de ces développements. De plus, cette approche offre des bénéfices multiples : la mise en place de la simulation sur les plateformes d'exécution en EE peut être en partie automatisé, une meilleure intégration de la simulation dans le processus de développement système, une traçabilité explicite entre le modèle d'architecture du produit et les éléments de simulation. De ce fait, il en découle une réduction des erreurs d'architectures système du « Produit », des spécifications vers

la couche physique « Equipements » plus abouties (Phase 3, figure 3.1), un gain de temps dans la conception du produit. Par exemple, ces erreurs d'architecture peuvent porter sur une mauvaise estimation du dimensionnement des échanges de données entre le système et le monde extérieur, où encore sur l'absence de prise en compte d'événements telle qu'une alarme.

3.2.3 Une approche matricielle

Cette approche matricielle représente d'un côté les acteurs (indépendants entre eux, conformément au premier principe fondateur) et de l'autre, les étapes de modélisation de l'architecture du produit, basées modèles (deuxième principe), orthogonales à ces mêmes acteurs. L'approche matricielle est intéressante pour avoir une vision d'ensemble de cette méthode (figure 3.4).

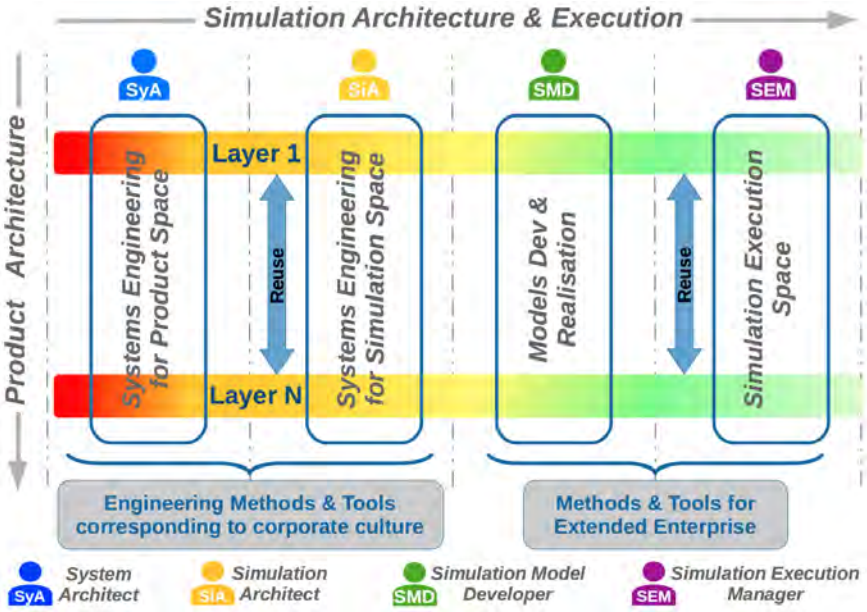

FIGURE 3.4 – Méthode générique, architecture matricielle.

Les sections suivantes décrivent les axes verticaux puis horizontaux, de cette matrice.

3.2.3.1 L'espace ingénierie système pour le "Produit"

Dans cet espace d'ingénierie système, point d'entrée de la méthode, l'objectif de l'architecte système du produit (SyA) est de concevoir l'architecture du produit répondant aux exigences de besoin de son client, de rédiger les exigences pour développer les équipements

et leurs intégrations. Suivant le premier principe fondateur de la méthode, il est indépendant des autres acteurs. À ce titre, il a le choix d'utiliser la méthode d'ingénierie système, basée modèle, qui lui convient le mieux, ou encore d'adopter la méthode prônée par son entreprise ou sélectionnée par les partenaires de l'entreprise étendue.

3.2.3.2 L'espace ingénierie système pour la simulation

Cet espace d'ingénierie système pour la simulation est animé par un nouvel acteur : l'architecte système pour la simulation (SiA). À l'identique de l'architecte système pour le produit (SyA) et conformément au principe premier de la méthode, il est indépendant des autres acteurs et peut appliquer sa propre méthodologie MBSE. Son objectif est de concevoir son propre produit, à savoir, une architecture de simulation.

L'architecte système pour la simulation (SiA) a pour rôle de faire le lien entre l'architecte système du produit (SyA), les développeurs des modèles de simulation (SMD) des différents domaines métiers (sécurité, sûreté, mécanique, thermique, etc.) et des personnes en charge de la préparation et de l'exécution de la simulation (SEM). Le SiA se doit d'avoir une culture technique assez large, et de bien connaître les spécialités métiers des différents partenaires.

Parmi ses objectifs, le SiA cherche à réduire les temps de simulation, lorsque cela est possible, pour au choix : réaliser plus de simulations en un temps donné, ou encore, réduire les coûts associés à l'utilisation de la plateforme de cosimulation. À cette préoccupation première, il peut être amené à regrouper, séparer des fonctions, à modifier l'architecture initiale du produit en provenance du SyA (toujours avec son accord), pour tenter de minimiser les impacts sur la qualité des données lors des interpolations, lors de l'introduction de retard entre deux fonctions. Pour finir, il dresse l'architecture de simulation, par allocation des modèles de simulation à destination de chaque partenaire. Il intervient sur l'algorithme de cadencement de la simulation, pour en préciser le pas de simulation, prendre en compte les boucles de simulation et pour s'assurer que les modèles ne divergent pas. Il s'appuie, également, sur les fonctions support de la plateforme de cosimulation, pour, par exemple, sélectionner et réutiliser des modèles de simulation déjà réalisés, afin de préparer de nouvelles simulations. En dernier ressort, l'architecte pour la simulation fournit les exigences à destination des développeurs des modèles exécutables de simulation et aux personnes en charge de la préparation et de l'exécution de la simulation. Ces exigences sont détaillées à la section 3.2.3.6 et illustrées par la figure 3.6 (p. 75).

3.2.3.3 L'espace des développeurs de modèles

Le rôle des développeurs de modèles existe depuis bien longtemps. Durant l'antiquité, il était possible de trouver des développeurs d'astrolabes. Celles-ci servaient à mesurer la hauteur d'un astre et à déterminer l'heure de l'observation [Wik19a]. Plus tard sont apparues les sphères armillaires, dont une de ces sphères stylisées apparaît sur le drapeau portugais.

Quant aux développeurs informatiques des modèles de simulation (SMD : Simulation Model Developper) d'aujourd'hui et dans le cadre de cette proposition, ceux-ci doivent réaliser les modèles exécutables suivant les exigences spécifiées par le SiA (section 3.2.3.6, figure 3.6). Il peuvent, à ce titre, se baser sur leur propre méthodologie système pour produire l'architecture de leur modèle, puis réaliser le modèle exécutable de simulation. Pour préserver la confidentialité des modèles, le temps de calcul ou encore le coût d'utilisation de la plateforme de cosimulation, le développeur est amené à développer le modèle de simulation sur sa propre plateforme. Par la suite, le modèle exécutable produit est chargé sur la plateforme de simulation de l'entreprise, pour réaliser les tests de déploiement. Ce modèle est enfin intégré avec les autres modèles sur la plateforme de cosimulation pour vérifier et valider les échanges des données entre les modèles des autres partenaires. En agissant ainsi, c'est-à-dire en ne déployant que le code exécutable sur la plateforme de cosimulation, le développeur protège de facto et au premier niveau le savoir-faire de son entreprise. Bien évidemment, il est toujours possible, en second niveau, de chercher à faire de la rétro-ingénierie des exécutables, pour essayer d'appréhender le savoir-faire de l'entreprise réalisant le modèle de simulation, source de l'exécutable.

À charge pour le développeur de respecter le document méthodologique du projet, qui précise les contraintes à respecter. Parmi ces obligations, il est possible de citer le respect de l'API de simulation, de l'encodage des données (Big/Little Endian) pour leur transfert sur le réseau, de la fourniture des différents renseignements, constituant la carte d'identité du modèle exécutable de simulation : le numéro de version, la date de réalisation, le nom de la société et de l'auteur, la liste des exigences associées pour l'aspect traçabilité, un texte donnant une explication succincte de ce que fait le binaire (par ex, 2000 caractères Max), la référence du document associé pour avoir plus d'informations sur ce binaire, et tout autre élément utile.

3.2.3.4 L'espace pour l'exécution de la simulation

Enfin, il est indispensable qu'au sein de chaque entreprise partenaire, un acteur (SEM : Execution Simulation Manager) soit en charge de l'exécution de la simulation, et ceci, de par la nature même de cette entreprise étendue. Même si cet acteur existe déjà dans les faits,

il apparaît à travers les témoignages des membres de l'équipe MOISE qu'il faut donner un périmètre d'intervention et un statut clair et reconnu à cet acteur. Ce « nouveau » rôle est important et multiple. Ce « nouvel » acteur est chargé de recevoir les modèles de simulation exécutables, d'assurer une gestion des versions et de la documentation associée, d'installer les exécutables, et de mettre en configuration de lancement la plateforme de simulation. Pour ce faire, l'acteur SEM s'appuie sur les outils de gestion des configurations proposés par la plateforme de simulation. Le SEM est également l'interface avec le développeur des modèles, pour lui fournir l'environnement adéquat pour le portage, les tests unitaires et d'intégrations des modèles de simulation.

3.2.3.5 Fonction prescriptive et environnement descriptif

La section précédente précise le rôle des acteurs, colonne vertébrale de la matrice (figure 3.4). Toutefois, il est nécessaire de préciser les qualificatifs « prescriptif » et « descriptif » qui sont associés aux sémantiques des fonctions et des environnements, avant d'aborder dans la section suivante, les échanges de données entre les différents acteurs.

Dans la langue française, parmi les différents sens qu'il est possible d'accorder à l'adjectif *prescriptif*, il en est un qui signifie : « *Qui constitue un commandement* » [CNR18]. Dans le cadre de cette proposition, ce *commandement* correspond, dans les faits, à une exigence. Le dictionnaire Larousse [Lar18] donne plusieurs définitions pour le terme *descriptif*. La première correspond à celle-ci : « *Qui s'attache à décrire quelque chose pour produire un effet artistique, recréer une ambiance, etc.* ». Dans le contexte de cette proposition, il est possible d'exprimer le terme *descriptif* par : « *Qui s'attache à décrire, à base de modèle, une approximation de la réalité, pour produire l'effet escompté* ».

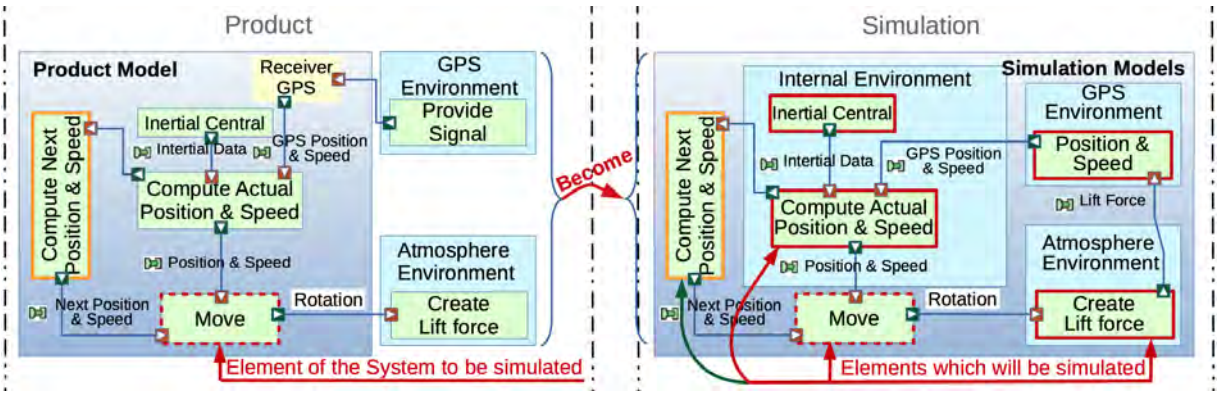

FIGURE 3.5 – Functional architecture & associated simulation model.

De nombreux auteurs (e.g. [Lee16]) arguent qu'il existe deux types fondamentaux de modèles : les modèles scientifiques qui décrivent le comportement des systèmes observés et

les modèles d'ingénieries (Product Model) qui prescrivent le comportement attendu d'un système. Ces deux types de modèles doivent être aussi simples que possible en regard de l'objectif de la simulation. Ils peuvent être de type : continu, échantillonné ou hybride. [Lee16] suggère que ces types de modèles ne doivent pas être traités de la même manière. Les modèles d'ingénierie du produit (Product Model) doivent être *prescriptifs*, tandis que les modèles d'environnement sont *descriptifs* et doivent servir à évaluer les modèles du produit.

En prenant appui sur les suggestions de Lee, dans les espaces du produit (figure 3.5 : Product) et de la simulation (figure 3.5 : Simulation), les modèles d'environnements des fonctions GPS Environment et Atmosphere Environment semblent pleinement appartenir à la catégorie des modèles scientifiques, dans la mesure où ils doivent décrire les comportements des systèmes observés (propagation d'ondes, condition atmosphérique). À ce titre, ces modèles sont de type *descriptif*.

Dans l'espace du produit (Product), l'ensemble des fonctions sont prescriptrices, puisque ces modèles de fonctions décrivent le comportement du produit final attendu : Receiver GPS, Inertial Central, Compute Actual Position & Speed, Move et Compute Next Position & Speed.

Toutefois, ce n'est pas toujours le cas pour ces mêmes fonctions dans l'espace de simulation. Toujours dans cet exemple de la figure 3.5 et pour donner le contexte, la fonction Move (cerclée en rouge pointillé) est au centre de l'attention et doit être simulée, pour en vérifier son comportement. Elle est de facto prescriptrice. La fonction Compute Next Position & Speed, cerclée de couleur orange, est une fonction qui a déjà été simulée. Elle est, ici, réutilisée et est également prescriptrice.

Dans l'espace de la simulation et pour les fonctions Inertial Central et Compute Actual Position & Speed, leur statut est particulier. Elles conjuguent les deux natures : elles sont à la fois *prescriptives*, de par leur origine et *descriptives* de par leur rôle d'environnement vis-à-vis de la fonction Move. L'aspect prescriptif concerne plus particulièrement le flux de données entre la fonction Compute Actual Position & Speed et la fonction Move. Ce flux de données doit respecter l'interface d'entrée de la fonction Move. À charge pour l'architecte de la simulation de *décrire* le comportement des fonctions Inertial Central et Compute Actual Position & Speed de manière à réduire les temps de calcul et à conserver le sens global.

Maintenant que les définitions des fonctions *prescriptive* et *descriptive* sont données, il est plus facile d'aborder la section suivante qui concerne les échanges de données entre les différents acteurs.

3.2.3.6 D'un acteur à l'autre : du modèle du produit à la simulation

Cette section précise plus concrètement le cheminement nécessaire pour obtenir un modèle de simulation à partir du modèle du produit (figure 3.2).

Le parcours du modèle « Produit » vers la simulation (figure 3.6) se veut générique, quel que soit le nombre de lignes constituant la matrice (figures 3.4) et quelle que soit la ligne considérée de cette même matrice.

L'architecte système du produit reçoit les exigences de la part de son client. Les exigences clientes sont traduites en scénarios de tests modélisés ou non.

L'architecte système du produit (SyA) cherche à s'assurer du bien fondé de son architecture. Pour ce faire, il fournit, à destination de l'architecte en charge de la simulation (SiA), les éléments nécessaires et suffisants pour répondre à son objectif de simulation (figure 3.6). Ces éléments sont, entre autres :

- le modèle de l'architecture,
- l'objectif de la simulation pour pleinement informer les autres acteurs quant aux buts de cette demande,
- les exigences (propres à l'architecte système de produit (SyA)) et leurs références, sous la forme de scénarios pour les tests unitaires et d'intégration,
- les scénarios des tests, correspondant aux exigences d'expression du besoin du client,
- les références des exigences associées aux spécifications des besoins de son client, pour la traçabilité de bout en bout,
- les sémantiques prescriptives des fonctions constituant le modèle d'architecture et les sémantiques descriptives de l'environnement,
- les exigences, de l'architecte système de produit (SyA), quant à la visualisation de l'évolution de certaines données de simulation (paramètres d'entrées/sorties des fonctions système, des fonctions d'environnement).

L'architecte système du produit doit exprimer, sans ambiguïté, sa pensée quant au sens qu'il donne précisément aux différentes fonctions de son modèle d'architecture. Ces sémantiques prescriptives sont considérées comme exactes, par nature, par les autres acteurs. Parmi la variété des possibilités qui s'offre à lui, il peut s'appuyer, de manière non exclusive, voire inclusive, sur une description textuelle, une description algorithmique, un modèle réalisé sous forme de code (ex : C++, Fortran, etc.), un modèle à base d'un outil dédié (ex : Modelica, Simulink, etc.), comme illustré par la figure 3.6.

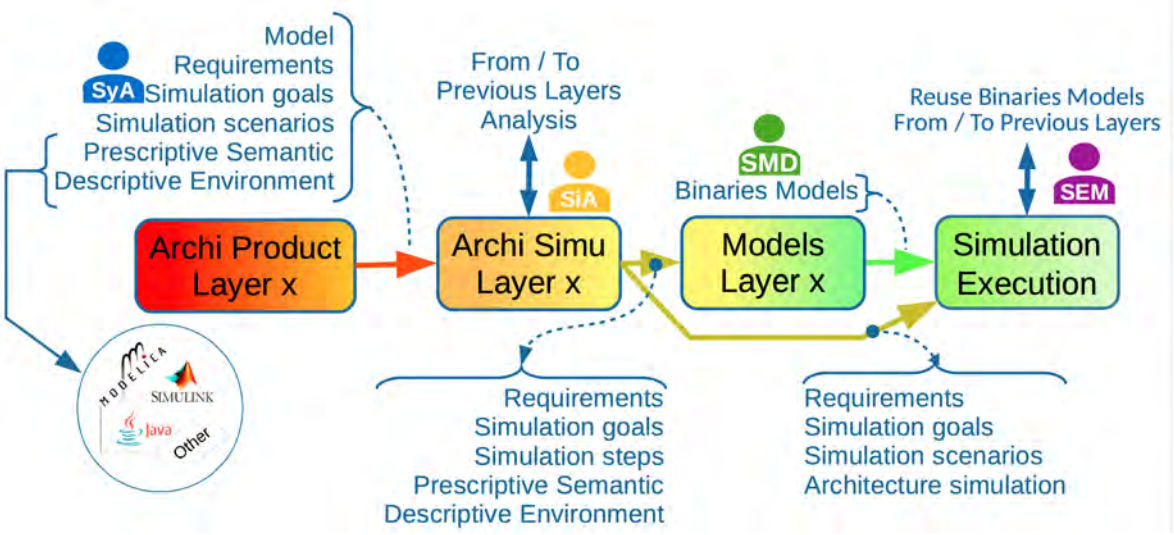

FIGURE 3.6 – Layer x : du modèle « Produit » à la simulation.

Si la sémantique prescriptive ne doit pas souffrir d'ambiguïté, il en est de même pour la sémantique descriptive de l'environnement du produit. La sémantique descriptive s'attache à préciser les comportements de l'environnement du produit, nécessaires et importants pour une simulation particulière, sans pour autant décrire complètement cet environnement. En effet et par exemple, est-il nécessaire de modéliser l'atmosphère dans sa complexité, lorsque seules la direction du vent et sa force suffisent, pour un objectif de simulation ? Autre exemple, est-il impératif de modéliser l'ensemble des interactions dans un aéroport, lorsque seule l'autorisation d'envol est nécessaire à la simulation ? Bien évidemment, les outils, à utiliser pour spécifier la sémantique descriptive, peuvent être identiques à ceux employés pour décrire la sémantique prescriptive (figure 3.6).

L'architecte de la simulation (SiA) reçoit l'ensemble des exigences, expression du besoin de l'architecte du « Produit » SyA. Sur cette base, le SiA applique sa propre méthode d'ingénierie système, basée modèle, pour définir son produit : l'architecture de simulation. Pour construire son architecture économe en temps de simulation, il recherche, dans les modèles fournis par l'architecte du « Produit », les fonctions pouvant être considérées comme chronophages et n'apportant pas une réelle plus-value pour l'objectif de la simulation. Dans ce cas et en accord avec l'architecte du « Produit », il simplifie les fonctions chronophages, et au besoin, il modifie par exemple certaines parties de l'architecture du modèle : en regroupant des fonctions dans un élément composite, en scindant des fonctions, en supprimant un niveau composite. De plus, l'architecte de la simulation prend en compte les demandes particulières de l'architecte du produit, telles que la visualisation de l'évolution de certains paramètres de simulation. Pour ce faire, le SiA ajoute à l'architecture de simulation, les fonctions nécessaires pour cet objectif de visualisation. En fonction des demandes de l'architecte du produit, l'architecte de la simulation peut y ajouter des éléments

de perturbation des données. Si la simulation l'exige, notamment dans le cas d'horloges dont les cadencements sont différents, le SiA peut adjoindre des fonctions de lissage ou d'interpolation, en amont ou aval de certaines fonctions.

La méthode proposée s'appuie sur le modèle du Produit pour créer le modèle de simulation et minimiser les écarts potentiels entre ces deux modèles (figure 3.2). En plus de cet avantage, cette méthode offre des gains substantiels tant en termes de temps qu'en coût de développement. De plus, elle permet de conserver, créer, maintenir des liens de traçabilité lors des raffinements (figure 3.7) ou des regroupements (figure 3.8) de fonctions. Ces gains sont réalisés par la réutilisation des fonctions déjà définies, développées et validées. Cette réutilisation prend deux formes : réutilisation « Amont » ou « Aval ».

Dans le premier cas (figure 3.7), la réutilisation « Amont » signifie que l'architecte de la simulation réemploie une ou des fonctions de la couche `Step x-1` (ici, les fonctions `Fct A` et `Fct N`) lorsqu'il travaille sur la couche `Step x`. Cela peut être par exemple, lorsque l'architecte du produit précise, dans la couche `Step x` un raffinement ou un choix technologique sur une fonction particulière (`Fct P1` et `Fct P2`), avec réutilisation des fonctions de la couche précédente (`Step x-1`) pour vérifier que son choix technologique répond toujours aux exigences de besoin.

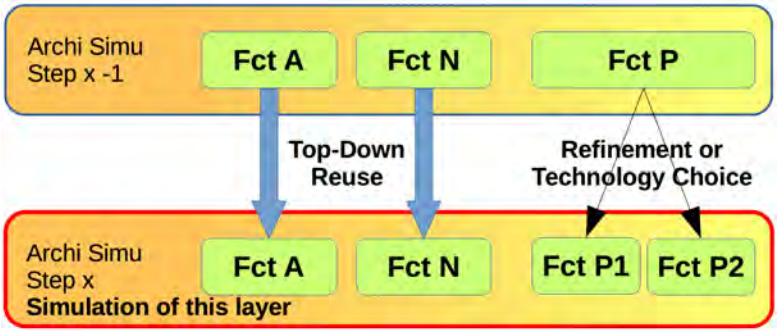

FIGURE 3.7 – Espace architecture de simulation : Réutilisation « Amont ».

En repartant de l'exemple précédent, l'architecte du produit modifie/ajoute des fonctions dans la couche `Step x-1` (figure 3.8, fonctions `Fct C` et `Fct Y`). Il peut vouloir vérifier cette nouvelle architecture en réutilisant les fonctions technologiques, précisées dans la couche `Step x` (fonctions `Fct P1` et `Fct P2`). La réutilisation de ces fonctions technologiques est vue comme une réutilisation « Aval ».

À noter que dans cette description, la couche `Step 1` correspond à la première couche de décomposition système (figure 3.4). Cette couche est généralement associée à la modélisation de l'expression des besoins du client. La dernière couche dénommée `Step N` peut être associée au dernier raffinement de la couche système dite « Physique », ou encore, « Construction » pour la méthode CESAM [CES19].

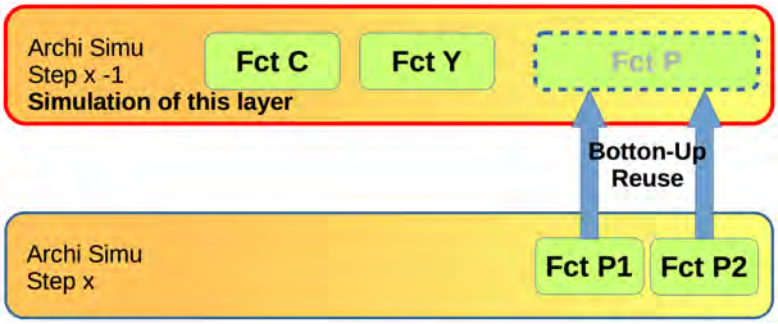

FIGURE 3.8 – Espace architecture de simulation : Réutilisation « Aval ».

Lorsque le simulateur le plus adapté aux besoins du client par minimisation de certains critères de coûts (par exemple, réduction potentielle du temps de simulation) est défini, lorsque les réutilisations « Amont » ou « Aval » sont précisées, lorsque les demandes additionnelles sont ajoutées, l'architecte de la simulation (SiA) affecte, dans la couche « Physique » de sa décomposition système (figure 3.17), les différentes fonctions sur les modèles des unités d'exécution des partenaires, en accord avec leurs spécificités métiers. Puis, le SiA génère les codes des interfaces de communication, les squelettes de l'orchestration de la simulation, les codes nécessaires à la visualisation des paramètres, les informations utiles à destination des développeurs des modèles de simulation exécutables (SMD), et en direction des responsables pour l'exécution de la simulation (SEM) (figure 3.6). À charge pour les développeurs des modèles de simulation, sur la base des exigences d'entrées (les exigences, le but de la simulation, les sémantiques prescriptives, descriptives, le pas de simulation), de fournir les exécutables adaptés à chaque partenaire, en fonctions de la répartition déterminée par l'architecte de la simulation (SiA) et d'y ajouter les éléments nécessaires à la traçabilité.

Dans chaque entreprise partenaire, le responsable de l'exécution (SEM) de la simulation reçoit les exigences d'entrée et les exécutables qui lui sont destinés. Il installe les exécutables et met en configuration sa plateforme de simulation, en fonction des exigences d'entrée en provenance de l'architecte de la simulation, pour la prochaine simulation en entreprise étendue.

3.2.3.7 D'un acteur à l'autre : le métamodèle

La section précédente a décrit le flux des données et le transfert des exigences, entre les différents acteurs. Cette section décrit le métamodèle sous-jacent, utile à la réalisation de ces transferts, de cette traçabilité entre exigences et de la composition des fonctions.

Ce métamodèle (réalisé sous Eclipse/Ecore) se base sur quatre grands concepts : les concepts de `Transfert`, d'`Exigence`, d'`Acteur`, et de `Modèle`.

Le concept de Transfert est le point d'entrée de ce méta-modèle. Il contient en attribut, l'objectif de la simulation butSimulation. Il est composé par un ensemble d'exigences ([1..*]exigence), de l'ensemble des acteurs partie prenante ([5..5]acteur). Ce concept précise également le sens de transfert des données [1..1]de et [1..1]vers entre les acteurs. Sur ce sens de ce transfert, il est précisé, dans l'objet commentaire, les contraintes à appliquer :
- de l'ArchitecteDuProduit vers l'ArchitecteSimulation,
- de l'ArchitecteSimulation vers le DeveloppeurSimulation,
- de l'ArchitecteSimulation vers le ResponsableExecution, et
- du DeveloppeurSimulation vers le ResponsableExecution.
La métaclasse Transfert possède un dernier lien de composition avec la métaclasse Sémantique_Modèle. Ce concept, après instanciation, peut représenter, par exemple, aussi bien un diagramme fonctionnel, comme un diagramme activité ou tous autres diagrammes signifiant une dynamique d'exécution.

Le concept d'Acteur traduit les différents intervenants, tels qu'énoncés dans les sections précédentes. Le choix de typer les acteurs a été fait dans le but de pouvoir faire des analyses en partant des types et non sur un quelconque attribut textuel associé à la métaclasse Acteur, ou encore, lors de l'instanciation, d'imposer la sélection non équivoque des types d'acteurs, pour spécifier le sens du transfert des données. Cette métaclasse spécialise les acteurs ArchitecteDuProduit, ArchitecteSimulation, Developpeursimulation, ResponsableExecution et Contexte. Ici, Contexte signifie l'ensemble des parties prenantes externes au système (personne, système, lois, standards, etc.). Cela permet de préciser l'origine Contexte pour les exigences extérieures au système. Les acteurs possèdent également une référence ([0..*]possède]) vers les exigences qu'ils peuvent produire. Cette référence facilite, également, la navigation dans le métamodèle.

L'introduction de la métaclasse Exigence permet de prendre en compte l'ensemble des exigences associées à l'objectif de la simulation. Cette métaclasse possède deux attributs : une référence d'identification textuelle et la description textuelle de l'exigence. Le lien de référence [1..1] créePar permet d'en préciser son auteur. L'autre référence importante, c'est le lien [0..*] utiliséPar. Ce lien permet de rechercher les fonctions et les sémantiques impactées par l'exigence.

Les références [0..*] amont et [0..*] aval assurent la traçabilité entre exigences. Enfin, le but du lien [1..1] possédéPar est de faciliter la navigation dans le métamodèle.

Les métaclasses Modèle, Function et Sémantique forment un double patron de conception « Composite ». Le premier patron de conception « Composite » est constitué par la métaclasse Sémantique qui est l'élément « feuille » de l'élément composite Function, avec la contrainte qu'une fonction possède une seule et unique sémantique. L'intérêt de ce premier patron de conception est de pouvoir définir plusieurs sous-fonctions, au sein d'une

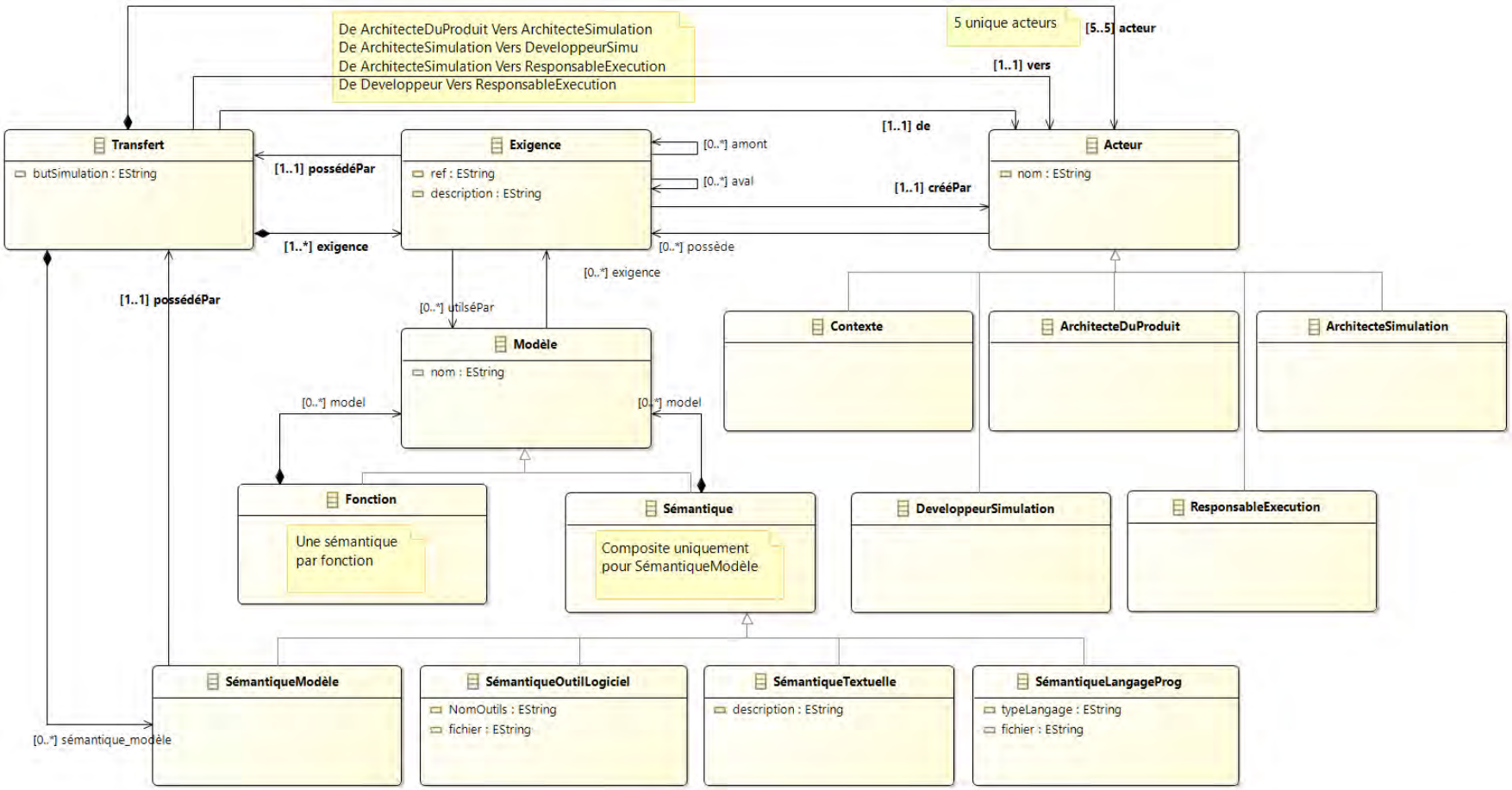

FIGURE 3.9 – Méta-modèle pour le transfert des données entre les acteurs et la traçabilité

même fonction. Dans ce cas, la fonction englobante ne possède pas de sémantique attachée. Des sémantiques sont alors assignées à chacune des sous-fonctions. Le deuxième patron de conception « Composite » est définit par les trois métaclasses présentées précédemment, où la métaclasse `Sémantique` constitue l'élément « Composite » et la métaclasse `Function` en est la « feuille ». Ce patron ce conception prend tout son sens pour la sémantique dénommée `Sémantique Modèle`, et uniquement pour celle-ci. En effet, une `Sémantique Modèle` peut être aussi bien un diagramme d'activité, un diagramme fonctionnel, ou tous autres modèles graphiques décrivant le comportant d'une fonction. Dans ce cas, le patron de conception « Composite » permet de définir les fonctions associées à ce `Sémantique Modèle`. Le lien `[1..1] possédéPar`, au départ de la métaclasse `Sémantique Modèle`, facilite le retour à la métaclasse `Transfert`. La métaclasse `SémantiqueOutilLogiciel` signifie que la sémantique d'une fonction est donnée par un modèle logiciel, par exemple, de type Modelica, Simulink, ou tous autres logiciels de modélisation comportementale. A cette métaclasse est attachée deux attributs, l'un pour spécifier le logiciel, l'autre pour indiquer l'emplacement du fichier. Comme son nom l'indique, la métaclasse `SémantiqueTextuelle` fournit une description textuelle d'une sémantique, tandis que la métaclasse `SémantiqueLangageProg` précise une sémantique via un algorithme défini par un langage de programmation.

3.3 Présentation de la plateforme de cosimulation

Un des objectifs du projet MOISE est de réaliser des simulations en entreprise étendue (EE). L'architecture générale de simulation se doit donc d'intégrer cette notion d'entreprise, auquelle s'ajoute les unités d'exécution de chacune d'entre-elles, ainsi que les moyens de communication entre les différents partenaires. Une illustration en est donnée par la figure 3.10. Cette architecture générale de simulation est dénommée par la suite : plateforme de cosimulation en entreprise étendue, ou plus simplement plateforme de cosimulation.

À notre connaissance, il n'y a pas été trouvé, dans la littérature, de modèle de référence de plateforme de cosimulation. L'objet de cette section contributive tente d'extraire les éléments essentiels constitutifs de cette plateforme de cosimulation, afin de déterminer une communalité entre les plateformes des différents partenaires. Ici, le MBSE (Analyses opérationnelle, fonctionnelle et physique) est mis en œuvre tant pour la démarche systémique que pour répondre à cette tentative.

Sur la base de ce premier objet, il s'y ajoute la détermination d'un métamodèle pour prendre en compte les diversités d'architecture que peuvent prendre les plateformes de cosimulation en entreprise étendue. Enfin, parmi les résultats de cette démarche et en plus des acteurs définis dans la section précédente, il apparaît utile de définir trois nouveaux

acteurs : le IPM (Infrastructure Project Manager), un responsable IPM et un responsable SEM, dont les rôles sont précisés dans la sous-section 3.3.1.

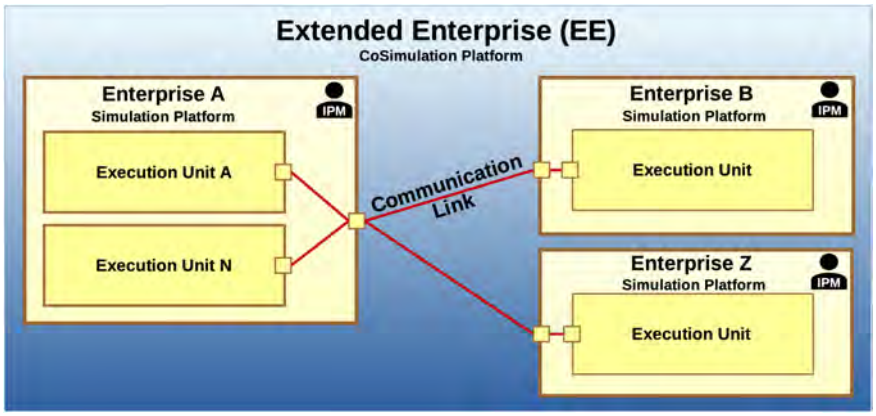

FIGURE 3.10 – Exemple d'une plateforme de cosimulation en Entreprise Étendue.

La figure 3.10 fixe l'idée générale de ce que peut être une plateforme de cosimulation en entreprise étendue, telle que définie par l'équipe MOISE. Elle est, par exemple, constituée par les entreprises A, B et Z. Pour réaliser les simulations, chaque entreprise possède une plateforme de simulation, constituée par une ou plusieurs unités d'exécution, reliées entre elles par des moyens de communication. L'ensemble des plateformes de simulation, en interrelation les unes avec les autres, constitue la plateforme de cosimulation en entreprise étendue. Or, construire cette plateforme de cosimulation n'est pas sans risque : coût et délais de réalisation, performance, coût d'exploitation. Pour chercher à minimiser les risques, la question est de savoir si les différentes plateformes de simulation peuvent avoir une structure commune. Une réponse peut être donnée en s'appuyant sur de bonnes pratiques pour le développement de systèmes complexes : par exemple, l'approche par l'ingénierie système basée modèle (MBSE).

Concomitamment à la mise en œuvre de l'une de ces méthodes MBSE pour concevoir la plateforme de cosimulation, il est possible d'avoir une démarche d'abstraction, pour isoler, puis extraire et réutiliser les idées et les facteurs communs nécessaires à la définition de l'architecture de la plateforme de cosimulation.

Les sections suivantes ont vocation à répondre à cette question, tout en posant les bases et les questions qui devraient aider à abstraire les éléments communs pour concevoir cette plateforme de cosimulation, au sein de l'EE. De plus, il est à noter que l'effet secondaire bénéfique de la présence de cette infrastructure de cosimulation est de pouvoir utiliser les moyens présents pour échanger, partager les modèles et les autres documents entre partenaires.

Dès le départ (section 3.3.1) et pour aborder/préciser la notion de plateforme de simu-

lation, il est important de définir les intervenants et leurs rôles, en interaction directe avec cette plateforme de cosimulation. La section suivante (3.3.2) s'attache à préciser l'expression des besoins (couche opérationnelle) du système. Cette section aborde les exigences d'architecture, de connexion réseau, de sécurité, de sûreté, les aspects logiciels et matériels et en dernier ressort les tests de validation/vérification de cette plateforme de simulation.

Sur la base des besoins exprimés, une architecture fonctionnelle de cette plateforme est donnée dans la section 3.3.3. Enfin, un modèle de plateforme de simulation peut être aisément représenté dans la couche physique (section 3.3.4, figure 3.12).

À ce stade de la démarche pour la définition d'une plateforme de simulation, les analyses systèmes MBSE sont réalisées. Il devient maintenant possible de chercher à abstraire des éléments pour les factoriser et les réutiliser. La section 3.3.5 décrit deux sortes d'abstraction pour décrire la structure de la plateforme de simulation. L'une aborde la notion de canevas, l'autre est basée sur un métamodéle, instancié dans la section 3.4.

3.3.1 Les acteurs de la plateforme de cosimulation

Dans l'environnement d'un système de simulation, il est possible de définir trois types d'acteurs, au sein de chaque entreprise partenaire.

Le premier type (nouveau) est dénommé *IPM (Infrastructure Project Manager)*. Ce type concerne les personnes qui sont en charge de la plateforme de simulation, au sein de leur entreprise. Leurs champs d'action sont particulièrement larges. Elles doivent s'assurer du respect des normes tant pour les personnes que pour le matériel (alimentation électrique, sécurité et sûreté, etc.) lors de la réalisation, puis du maintien en condition opérationnelle de la plateforme de simulation. Dans l'EE, leur rôle est essentiel. Elles devront dialoguer pour résoudre les problèmes administratifs entre partenaires, pour mettre en place les interconnexions informatiques externes et sécurisées, afin d'assurer le bon transfert des données lors de l'exécution des différents modèles de simulation. Les qualités requises sont, en autres : l'écoute, la médiation et des connaissances dans le montage et la maintenance des infrastructures informatiques matérielles et logicielles.

Le second type d'acteur est opérationnel : *SEM (Simulation Execution Manager)*. La personne, remplissant ce rôle au sein de son entreprise, est chargée de la préparation, du bon déroulement de l'exécution de la simulation, de la signalisation des problèmes d'exécution, etc. Son rôle a été plus amplement évoqué dans la sous-section 3.2.3.4. Elle est l'interface avec ses pendants chez les autres partenaires de l'EE.

Le dernier type d'acteur formalise la présence des développeurs des modèles de simulation : *SMD (Simulation Model Developper)*. Ils s'appuient sur cette plateforme de simu-

lation pour tester, intégrer et exécuter leurs modèles de simulation (voir la sous-section 3.2.3.3).

Les acteurs étant identifiés et leurs rôles précisés, il devient plus aisé de définir les exigences opérationnelles de ces acteurs. Ces exigences sont décrites dans la section suivante.

3.3.2 L'étape d'analyse opérationnelle

L'étape d'analyse opérationnelle recueille l'expression des besoins des partenaires, quant à la conception de cette plateforme de cosimulation. Ici, ces besoins sont axés sur la gestion de la plateforme en EE, sur la gestion des modèles de simulation et de la documentation, sur la sécurité, la sûreté et sur la nature des connexions entre les partenaires de l'EE.

Pour plus de lisibilité et catégoriser les besoins, tout en n'étant pas exhaustif, cette section est organisée par thème d'exigences. C'est également une contribution dans le sens où l'on s'appuie sur les pratiques constatées dans l'industrie et sur le MBSE, pour définir une plateforme de cosimulation.

3.3.2.1 Exigences centrées sur les acteurs

Chaque acteur doit pouvoir s'appuyer sur des fonctionnalités fournies par la plateforme de simulation, pour pouvoir remplir son rôle. Cette section liste de manière non exhaustive un certain nombre de ces exigences.

Comme exprimé plus haut, l'IPM (un par partenaire) est en charge de réaliser, au sein de son entreprise, la plateforme de simulation, puis d'en assurer le maintien en condition opérationnelle. À ce titre, il s'assure, directement ou par délégation, de la conformité aux normes en vigueur des locaux (par exemple, protection incendie) et de leurs aménagements, de l'adéquation des servitudes nécessaires aux personnels et aux matériels (ventilation, éclairage, téléphonie, etc.). Il gère les droits d'accès directs, et via le réseau, sur les unités d'exécution. Si cela est exigé par les standards de sécurité de son entreprise, il prend également en charge les droits d'accès aux locaux où sont installées les unités d'exécution (calculateurs).

Pour être opérationnel au sein de son entreprise, le responsable de l'exécution des simulations SEM a besoin de pouvoir sélectionner la version du modèle à exécuter et les potentielles IHM (Interface Homme-Machine) associées à la visualisation des paramètres internes aux modèles, de les charger dans l'espace d'exécution, de lancer les exécutables, de sauvegarder les traces d'exécution et autres fichiers générés dans le répertoire dédié, de retirer l'ensemble des exécutables.

Si les développeurs (SMD) des modèles de simulation peuvent réaliser une partie de leur travail sur leur propre unité d'exécution, ceux-ci vont devoir tester, vérifier puis valider unitairement leurs modèles, sur les unités d'exécution de la plateforme de simulation de leur entreprise. Pour ces tests de déploiement, ils devront pouvoir disposer des outils pour charger le code source de leurs modèles, le compiler, l'exécuter et apporter les potentielles corrections. Cela implique que les partenaires, lors de la définition de leur plateforme de simulation, devront intégrer cet aspect de la disponibilité des outils. Lorsque l'exécution de leurs modèles est conforme à leurs attentes, ils les sauvegarderont dans un espace dédié, soit au sein de leur société, soit en un autre lieu et dans un espace commun à l'ensemble des partenaires. Bien évidemment, il n'est pas exclu que l'unité d'exécution de chaque développeur soit également la plateforme de simulation de leur propre entreprise.

Suivant le type de management retenu : « hiérarchique » ou « réparti » pour réaliser l'architecture du produit ou pour produire l'architecture de la simulation, les fonctions associées aux acteurs de la plateforme seront quelque peu différentes. Dans une approche hiérarchique, deux nouveaux acteurs responsables apparaissent. Le premier supervise les IPM de chaque partenaire, pour piloter la réalisation globale de la plateforme de cosimulation, et pour le maintien en condition opérationnelle de celle-ci. Le second acteur responsable est chargé d'animer l'équipe des SEM pour la mise en œuvre et l'exécution de la simulation.

Ces deux nouveaux responsables, chacun dans leur domaine, devraient pouvoir interagir, à distance et de manière limitée, chez l'ensemble des partenaires. Le responsable des IPM devrait pouvoir lancer régulièrement, entre autres, des tests dévaluation des paramètres techniques de la plateforme de cosimulation (débit, temps de latence, etc), des tests de sécurité. Le responsable SEM, quant à lui, devrait pouvoir sélectionner l'ensemble des modèles, chez chaque partenaire, pour exécuter une simulation donnée, lancer les IHM[6] pour visualiser les paramètres souhaités.

Dans la vision management « réparti », les IPM se coordonnent et prennent les décisions de manière collégiale pour la mise en œuvre et la maintenance de la plateforme de cosimulation et de leurs propres plateformes de simulation. Le domaine d'intervention des IPM reste cantonné à leur entreprise. Il en est de même pour les SEM. À chacun d'œuvrer, pour préparer les modèles de simulation et de s'organiser pour leur exécution.

3.3.2.2 Exigences centrées sur l'architecture de la plateforme

Quelle architecture choisir pour la mise en place de cette plateforme de simulation : en étoile, sous forme d'arbre, en réseau maillé ? Qui définit cette architecture et quand ?

6. *Interface Homme-Machine*

Il est, bien sûr, possible d'obtenir une première indication sur cette architecture par la présence des partenaires et de leurs domaines d'activités. Par exemple, un grand avionneur mondial peut avoir tendance à faire le choix d'une architecture de type « hiérarchique » pour simplement avoir la maîtrise sur les différentes interfaces entre les constituants d'un avion. Toutefois, faire un choix anticipé peut être hasardeux, dans la mesure où il peut obérer certaines possibilités de simulation au niveau fonctionnel, par l'absence de relations entre certains domaines métiers. Cette architecture se doit également d'être évolutive et agile pour accueillir les entreprises entrantes et prendre en compte les entreprises sortantes.

In fine, il est utile d'attendre une première analyse fonctionnelle de haut niveau du produit, pour définir cette architecture. De cette manière, le coût d'installation des liaisons entre partenaires se fera au plus juste.

3.3.2.3 Exigences centrées sur les moyens de communication

Les choix du support physique et de la nature des connexions réseau entre partenaires sont des éléments cruciaux (fibre/câble/liaison satellite). Ce choix peut conditionner l'efficacité du travail, la sécurité, la vitesse d'exécution de la simulation, les coûts de mise en oeuvre, les coûts de location et d'utilisation.

Ces options ne sont pas exclusives entre elles. Tout dépend des performances demandées pour les échanges entre partenaires et de la confidentialité des données à échanger. Un partenaire pourrait très bien avoir plusieurs lignes spécialisées et une connexion Internet. L'utilisation de lignes spécialisées implique des coûts de mise en oeuvre, de location et d'exploitation supplémentaires.

Pour assurer l'interopérabilité entre les différentes architectures matérielles et logicielles utilisées par les partenaires, il devient impératif de préciser la manière dont seront codées les données (par exemple Little/Big Endian) pour assurer l'unicité des communications. Même si l'ensemble des OS [7], chez chacun des partenaires, est identique, il semble utile de préciser cet encodage des données, dans le cas ou une nouvelle entreprise arriverait en cours de projet.

3.3.2.4 Exigences centrées sur la sécurité/sûreté

Pour éviter les ambiguïtés possibles entre les mots de « sécurité » et « sûreté », [Piè10] aborde spécifiquement les relations entre ces deux termes, et en précise les définitions :

- La sécurité concerne les risques provenant de l'environnement ayant des impacts po-

7. *Operating System*

tentiels sur le système considéré [Piè10], par exemple : un acte de malveillance, une rupture de canalisation extérieure d'eau endommageant le système.

- La sûreté concerne les risques accidentels provenant du système et ayant des conséquences potentielles sur l'environnement [Piè10], par exemple : libération d'un gaz nocif ou la panne d'un calculateur.

La sécurité de la plateforme de simulation est multiforme : droits accès physiques dans les locaux, sur les unités d'exécution (les consoles), et protection des communications entre entreprises et entre unités d'exécution.

En particulier, avec une prévalence de plus en plus forte de virus, de chevaux de Troie, de « Malware » pour espionner, déstabiliser, rançonner les entreprises, et afin d'essayer de se prémunir de cette malveillance, la communication entre les entreprises partenaires du projet de simulation doit se faire via un réseau sécurisé. Suivant la nature et le degré de protection choisi, la sécurité de la plateforme de simulation peut se faire sur plusieurs niveaux.

La première approche consiste à sécuriser les communications entre les entreprises, via la mise en place de lignes de communication spécialisées et sécurisées, ou encore, via l'utilisation, par exemple de VPN [8] ou du standard TLS [9] (l'ensemble des standards SSL [10] ne sont plus supportés par l'IETF [11] [IET15]).

La deuxième approche concerne les droits d'accès à la plateforme de simulation, pour les locaux et les consoles des unités d'exécution (Identifiant / Mot de passe). Les questions concernant les droits d'accès sont quelque peu identiques au paragraphe précédent. Un droit d'accès unique ou un droit accès différent pour chaque partenaire ? À ce choix initial, faut-il définir des droits d'accès, donnant droit à des accès spécifiques, en regard des personnes et des fonctions associées ([AD03]) : droit d'accès des responsables de la plateforme (IPM), droit d'accès pour les personnes en charge de l'exécution des simulations (SEM), des personnes réalisant les modèles de simulation (SMD), droit d'accès pour le client, etc. ?

Pour la gestion des adresses IP et des ports de communication de la plateforme de simulation, la gestion se fait-elle sous forme centralisée ou répartie ? Comme pour d'autres questionnements, une partie des réponses sont données par le choix de la structure fonctionnelle de l'EE.

8. *Virtual Private Network*
9. *Transport Layer Security*
10. *Secure Sockets Layer*
11. *Internet Engineering Task Force*

3.3.2.5 Exigences centrées sur les logiciels & matériels de la plateforme

La plateforme de simulation nécessite des ressources tant matérielles que logicielles. Comment déterminer la quantité des ressources matérielles nécessaires à la simulation ? Ici, seule l'expérience professionnelle de chacun des partenaires permet de répondre à cette question. En effet, tout dépend de la nature de la simulation, du volume de l'objet à considérer, de la taille des mailles, des équations associées à chaque nœud. Suivant les cas, les puissances de calcul à mobiliser ne sont pas identiques.

Pour gérer les fonctionnalités de la plateforme de simulation, il y a un ensemble minimal de services qui peut être convenu dès le départ. Cet ensemble minimal peut être constitué par des logiciels assurant le contrôle du bon fonctionnement du réseau, le contrôle des accès, la gestion de la simulation (chargement des modèles, déchargement, départ, arrêt, mode pas-à-pas, retour en arrière), la gestion des versions des modèles de simulation et de la documentation, l'enregistrement des données de simulation et leur mise à disposition, au cours de la simulation, à l'ensemble des partenaires, suivant un format d'échange à spécifier dès le départ. Ces logiciels et leurs IHM associées sont considérés comme faisant partie de la plateforme de simulation. L'identification de ces éléments contributifs à la modélisation de la plateforme de cosimulation sont représentés dans la figure 3.11 par les éléments `Documentation Management`, `Models Simu Management` (`Simu` : abréviation pour simulation) et `HMI Management`. Ne pas confondre ces exigences logicielles avec les exigences, dédiées aux besoins propres des ingénieurs système du produit (e.g., IHM de suivi de l'évolution du mobile dans l'espace 3D, l'évolution des modèles, ou une partie de ceux-ci) et aux ingénieurs de simulation (détection des défauts de simulation). Ces logiciels, en effet, ne participent pas directement au fonctionnement de la plateforme de cosimulation.

3.3.2.6 Exigences centrées sur l'intégration de la plateforme

Les exigences centrées sur l'intégration de la plateforme de simulation précisent les procédures à réaliser pour vérifier, puis valider l'architecture de cette plateforme, les connexions réseau, la sécurité, la sûreté, les différentes fonctionnalités logicielles et matérielles associées à cette plateforme de simulation en EE. Dans une approche MBSE, il est utile de modéliser ces exigences. Or, ici, le gain ne semble pas suffisamment significatif et éclairant pour cette proposition de modélisation.

Pour bien faire la différence entre les éléments propres à la plateforme de simulation et les modèles de simulation, ces exigences ne prennent pas en compte l'intégration des modèles de simulation. Cette intégration, des modèles de simulation, fait l'objet de procédures spécifiques, hors de ce périmètre dédié l'intégration de la plateforme de cosimulation.

3.3.3 L'étape d'analyse fonctionnelle

Cette étape d'analyse fonctionnelle est construite à partir des exigences exprimées dans la section précédente. Le choix est fait d'adopter l'approche d'une entreprise étendue de type « Hiérarchique », dans le but de faire apparaître les responsables IPM et SEM (figure 3.11). Un second choix est réalisé pour séparer les services d'administration de la plateforme du service d'exécution de la simulation. Le but, ici, est de minimiser les "interférences" de la partie administrative lors d'une simulation.

Tout commence par l'acteur IPM (Infrastructure Project Manager). Il a en charge la création de la plateforme de simulation au sein de son entreprise (`Create Simulation Platform`). Via la fonction `Access Rights Management`, il définit les droits d'accès (`Define Access Rights`) pour les intervenants et les services : le SEM (Simulation Execution Manager), le SMD (Simulation Model Developer), les deux responsables IPM et SEM, sur les moyens informatiques, et sur les fonctions sensibles telles que les fonctions de protection contre les intrusions extérieures (`External Intrusion Protection`), les accès sur le gestionnaire du réseau de communication interne (`Network Management`), la configuration du répartiteur de données (`Route Data`), et les accès aux salles (`Access To Simulation Area`).

L'acteur SEM gère la préparation et l'exécution de la simulation sur les moyens d'exécution de son entreprise. Pour ce faire, il s'appuie sur le serveur de base offert par cette plateforme : `Server for Documentation & Models` pour accéder aux gestionnaires des services qu'il exploite : `Models Simu Management`, `Documentation Management` et `HMI Management`. Ces gestionnaires sont centrés sur la gestion du stockage, la gestion des versions des modèles de simulation et de la documentation associées, sur les IHM de contrôles/commandes et de visualisation des paramètres de simulation. Le service `Execution Simulation` assure l'exécution les commandes en provenance de l'SEM : Start / Stop / Step by Step, le séquencement de la simulation, la prise en compte des données d'entrées et le renvoi des données traitées, respectivement à partir puis vers le serveur de simulation.

Dans une vision MVC [12], `HMI Controller` est l'interface commune entre le serveur de simulation qui contient l'évolution des paramètres de simulation et les IHMs sélectionnées par l'acteur SEM. Cette interface se veut standard pour l'ensemble des IHMs qui viendront s'y connecter.

Deux rôles importants apparaissent dans une entreprise étendue de type « Hiérarchique » (cf section 3.3.2.1). Le premier, le responsable IPM (Infrastructure Platform Manager) s'assure du suivi quant à la réalisation de la plateforme de cosimulation, en lien avec les autres IPM des autres partenaires. Parmi ses fonctions et dans le but de protéger cette

12. *Model View Controller*

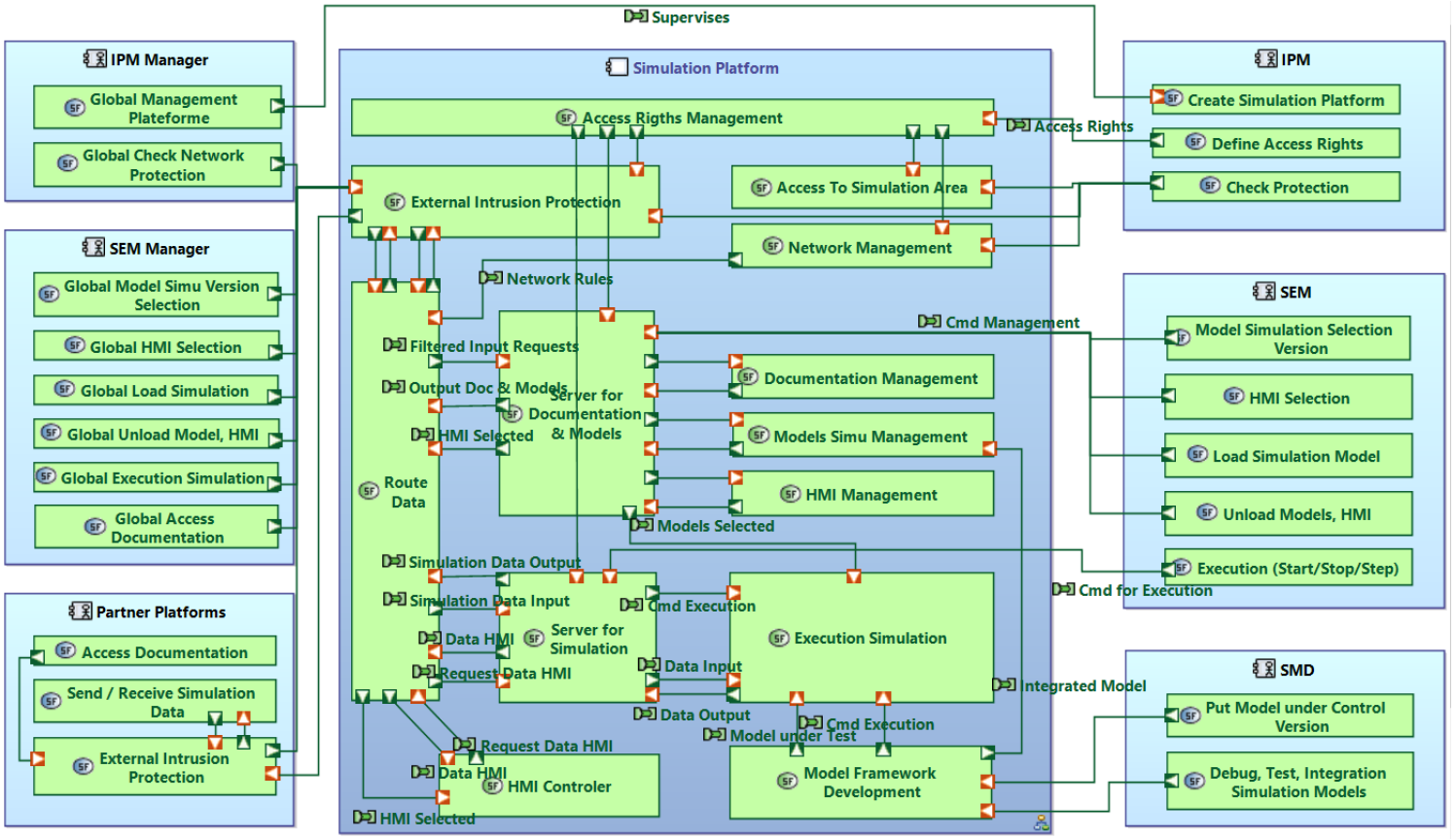

FIGURE 3.11 – Illustration de la couche fonctionnelle de la plateforme de simulation

plateforme de cosimulation de toutes intrusions malveillantes, il s'assure que les protections d'entrée de chaque partenaire sont bien efficientes (Global Check Network Protection).

Le second rôle, le responsable SEM, quant à lui, a l'aptitude de lancer une simulation complète à partir de son poste informatique, de manière manuelle ou automatique. Pour ce faire, chez chaque partenaire, il dispose des droits d'accès pour sélectionner (`Global Model Simu Version Selection`), charger (`Global Load Simulation`), exécuter (`Global Execution Simulation`), puis de retirer (`Global Unload Model, HMI`) les modèles de simulation. En lien avec cette simulation, il sélectionne les IHMs (`Global HMI Selection`) pour visualiser l'évolution temporelle des paramètres de simulation. Afin d'éclairer sa sélection, il accède à la documentation des modèles de simulation (`Global Access Documentation`). Par mesure de sécurité et pour limiter toutes tentatives de pénétrations malveillantes de la plateforme de cosimulation, l'ensemble des accès chez les partenaires se fait via la fonction de protection contre les intrusions externes.

Enfin, il reste les autres plateformes de simulation, constitutives de la plateforme de cosimulation (`Partner Platforms`). Elles ont accès à la documentation (`Access Documentation`). Elles fournissent/reçoivent (`Send / Receive Simulation Data`) les données de simulation, par l'intermédiaire de la fonction `External Intrusion Protection`.

Les fonctionnalités sont maintenant définies. L'objet de la section suivante « Définition du niveau physique » est de répartir ces fonctionnalités sur les unités d'exécution.

3.3.4 L'étape d'analyse physique

Dans la démarche d'ingénierie système basée modèle, cette étape d'analyse physique a vocation à préciser l'assignation des fonctionnalités, découvertes dans l'étape d'analyse fonctionnelle précédente, sur des équipements. Pour chaque partenaire, l'architecture physique est potentiellement différente. Cette analyse physique prend en compte, à la fois, les moyens disponibles et la spécificité métier des partenaires, pour le déploiement des fonctionnalités sur les moyens matériels. En effet, la répartition des fonctionnalités sera différente, si le partenaire ne dispose que d'une unité d'exécution, par rapport à un autre qui pourrait en mobiliser plusieurs. Les possibilités sont donc variées. Cela revient à dire que chaque projet collaboratif en entreprise étendue se doit d'être modélisé pour tenir compte de cette diversité.

Pour illustrer le propos, le choix est fait, ici sur un exemple d'instanciation (figure 3.12), de séparer la partie administration de la plateforme de simulation et la partie exécution de la simulation, sur deux unités d'exécution différentes. Ce choix est dicté par le souhait de ne pas « absorber » de la puissance de calcul au détriment du temps de simulation, lors, par exemple, des activités de gestion de la documentation, des modèles ou encore des

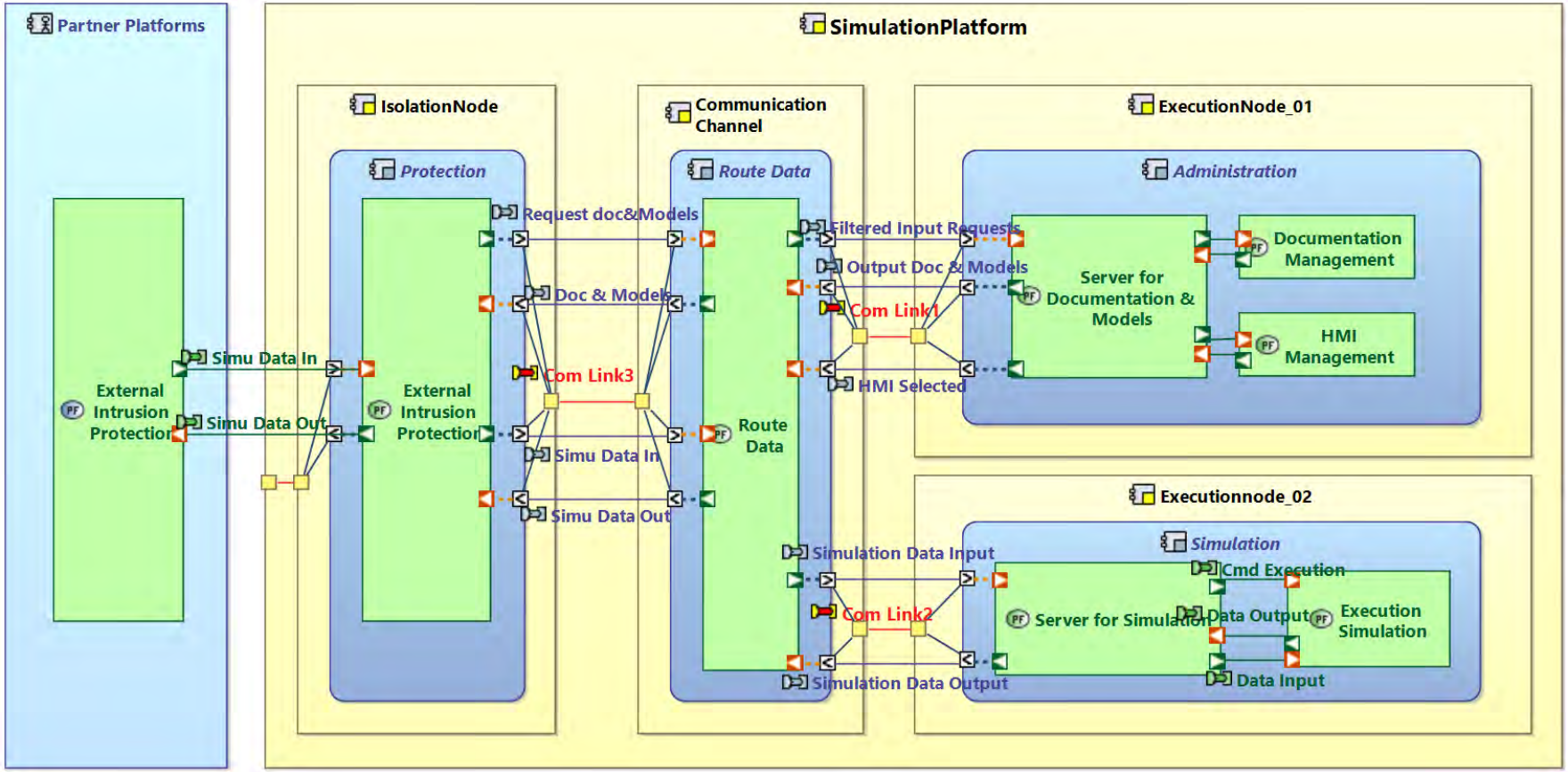

FIGURE 3.12 – Exemple d'instanciation pour une entreprise : les points essentiels de la couche physique

IHM. L'autre avantage de cette ségrégation est de pouvoir ajouter des unités d'exécution supplémentaires dans l'architecture.

Pour faciliter la lecture, sans dénaturer la compréhension, seules apparaissent, dans cette figure 3.12, les principales fonctionnalités et la nature de quelques données circulant sur les liens de communication.

Les résultats de l'analyse fonctionnelle (figure 3.11) montrent déjà, par la répartition géographique, les premières tendances d'assignation des fonctions à des équipements. Ici, le but est de répartir, de manière logique, la complexité d'ensemble sur des équipements de complexité moindre. L'intérêt, c'est la modularité, la possibilité d'assurer un support logistique et de maintenance plus efficace.

Le point d'entrée/sortie de la plateforme est constitué par la fonction External Intrusion Protection allouée au modèle de composant physique IsolationNode.

La fonction Route Data joue le rôle d'intermédiaire entre les communications de IsolationNode et les deux serveurs Server for Documentation & Models et Server for Simulation, ou encore entre les deux serveurs eux-mêmes. Sa fonction d'agrégation et répartition des données est suffisamment spécifique pour en faire un composant à part entière : CommunicationChannel.

De cette décomposition système fonctionnelle et physique, il est possible d'abstraire des concepts, d'extraire des idées communes qui peuvent être réutilisées chez chaque partenaire. La section suivante s'attelle à mettre en exergue ces communalités.

3.3.5 Abstraction

Deux approches coexistent, pour chercher à réutiliser des objets au sein de cette plateforme de cosimulation : a) La première consiste à extraire, à isoler pour abstraire une caractéristique d'un objet, ou encore, le caractère commun d'une collection d'objets. Cette démarche est réalisée à la section 3.3.5.1. b) L'autre se base sur l'extraction de propriétés, de cet ensemble d'objets, pour en faire des paramètres pouvant être instanciés. Cette méthode est décrite dans la section 3.3.5.2.

3.3.5.1 Abstraction par généralisation

En partant du concret (figure 3.13 : Activity V&V for one simulation), il est possible de dresser une expression des besoins communs à l'ensemble des partenaires (section 3.3.2). De cette analyse MBSE, basée sur les besoins exprimés, il en ressort, parmi l'ensemble des

fonctions identifiées, une architecture fonctionnelle pouvant être commune à l'ensemble des plateformes de simulation, telle que décrite par la figure 3.11. Elle peut être qualifiée d'« architecture fonctionnelle de référence ». Elle précise la structure, les liens que doivent avoir les fonctions entre elles, pour une plateforme de simulation, au sein de cette entreprise étendue. L'intérêt de cette approche est de pouvoir cloner cette structure (figure 3.13 : Generic Part Reuse), tout en laissant à chaque partenaire, la possibilité d'utiliser leurs propres outils. Ce dernier point est l'expression de l'indépendance des parties et de la préservation de la propriété intellectuelle sur les outils.

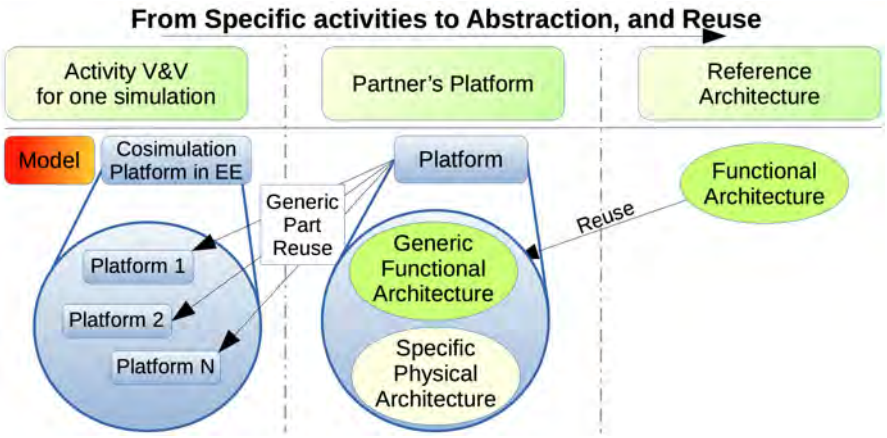

FIGURE 3.13 – Du spécifique vers le général (abstraction), puis la réutilisation.

Il est à noter que l'architecture physique peut être très différente d'une entreprise à une autre, de par le nombre d'unités d'exécution pouvant être mise en œuvre, ou encore, par l'organisation de leur système de communication interne. C'est la raison pour laquelle, il n'est pas spécifié, ici, d'architecture physique de référence, mais plutôt une architecture physique spécifique (Specific Physical Architecture).

3.3.5.2 Abstraction par métamodélisation

En s'appuyant sur les résultats issus de la décomposition système, basée modèle, d'une plateforme de cosimulation (couches fonctionnelle et physique), la création d'un métamodèle (fig 3.14) est une autre approche possible pour préciser les concepts qui pourront être instanciés afin de structurer et rendre homogène la partie fonctionnelle des plateformes de simulation, tout en prenant en compte la diversité des architectures physiques.

Le point de départ de ce métamodèle `IndustrialDomain` se veut large. C'est le contexte de cette modélisation. Il agrège les entreprises, les projets industriels `Project` de ce même domaine et le concept de canal de communication externe aux entreprises `WAN` [13].

13. *Wide Aera Network*

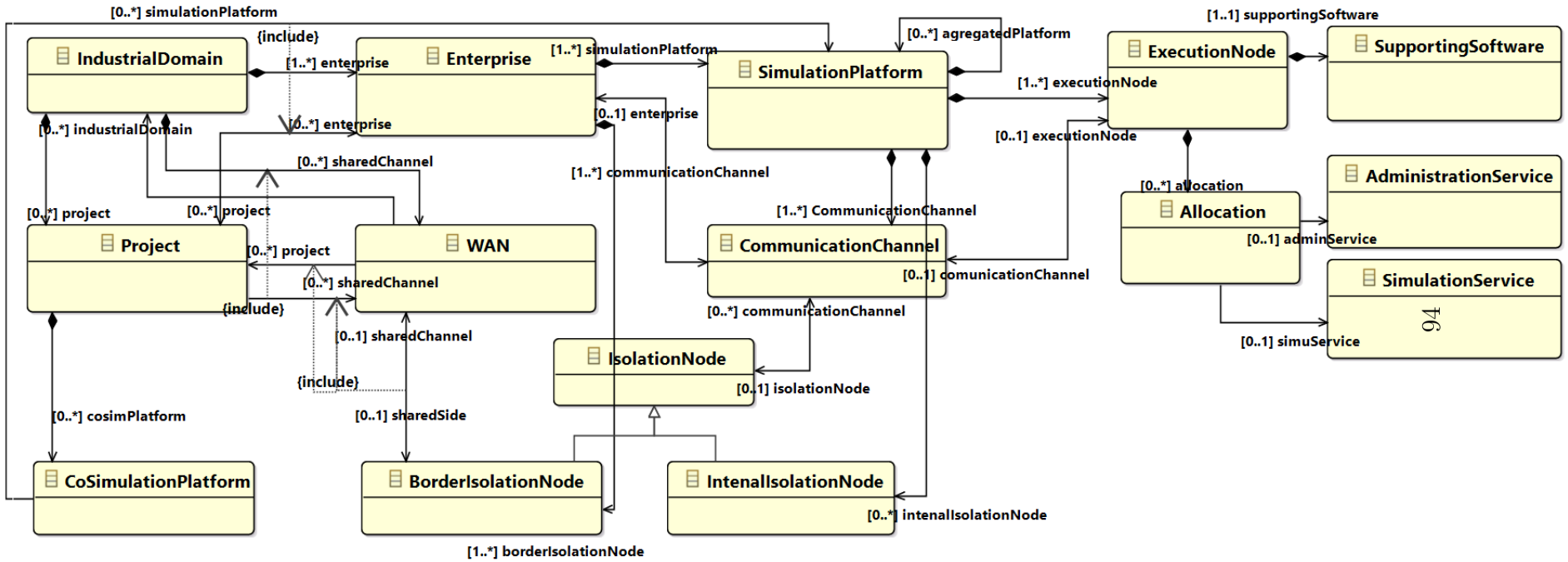

FIGURE 3.14 – Organisation des concepts d'une plateforme de cosimulation.

Si le Project appartient à un domaine industriel, il n'en est pas moins porté par des entreprises [0..*]entreprise. Il a la possibilité de communiquer avec les autres projets via WAN et sa relation au domaine industriel [0..*]industrialDomain. Pour être simulés, les projets s'appuient sur des plateformes de cosimulation CoSimulationPlatform qui agrègent les plateformes de simulation des entreprises impliquées dans ces projets industriels [0..*]simulationPlatform.

L'entreprise, quant à elle, possède un nœud d'isolation entre les mondes extérieur et intérieur IsolationNode, et un ensemble propre de plateformes de simulation SimulationPlatform, dont chacune agrège un ensemble de noeuds d'exécution ExecutionNode et au moins un canal de communication CommunicationChannel.

Le CommunicationChannel est le concept de communication central au sein de chaque SimulationPlatform, pour mettre en relation les nœuds d'exécution constituant la partie active de la plateforme de simulation. La plateforme de simulation peut aussi être une agrégation d'un ensemble de plateformes de simulation ([0..*]agregatedPlatform). Les échanges de données entre ces plateformes peuvent passer par un ensemble de CommunicationChannel dédiés et via des nœuds d'isolation interne InternalIsolationNode. Cette possibilité d'agrégation de plateformes offre à l'entreprise les moyens de mobiliser ses différentes ressources métiers, nécessaires à la cosimulation demandée par l'entreprise étendue. La lecture de l'exemple représenté par la figure 3.15 peut donner lieu à deux interprétations différentes. La première : les plateformes pour la simulation mécanique et hydraulique collaborent pour un même projet de cosimulation, via le canal de communication dédié à la simulation. La deuxième interprétation : ces deux plateformes réalisent chacune des simulations pour deux projets de cosimulation différents, d'où la présence des Internal Isolation Node.

Les communications de l'entreprise vers les partenaires passent par Communication -Channel, BorderIsolationNode [0..1]sharedSide, [0..1]sharedChannel WAN puis vers Project, et inversement. Les communications entre les différents ExecutionNode d'une même entreprise restent cantonnées au CommunicationChannel et aux Internal-IsolationNode, si ces derniers sont présents, puis les données à destination ou en provenance des partenaires passent par CommunicationChannel, BorderIsolationNode, et inversement.

De l'analyse fonctionnelle de la plateforme de cosimulation, il en émerge les services pour l'administration de la simulation, les services dédiés à l'exécution d'une simulation. Ces services sont représentés respectivement par les concepts : AdministrationService et SimulationService. L'exécution de ces services ne peut se faire sans un support matériel et logiciel. Le support matériel est représenté par le concept ExecutionNode, sur lequel tournent les services support SupportingSoftware.

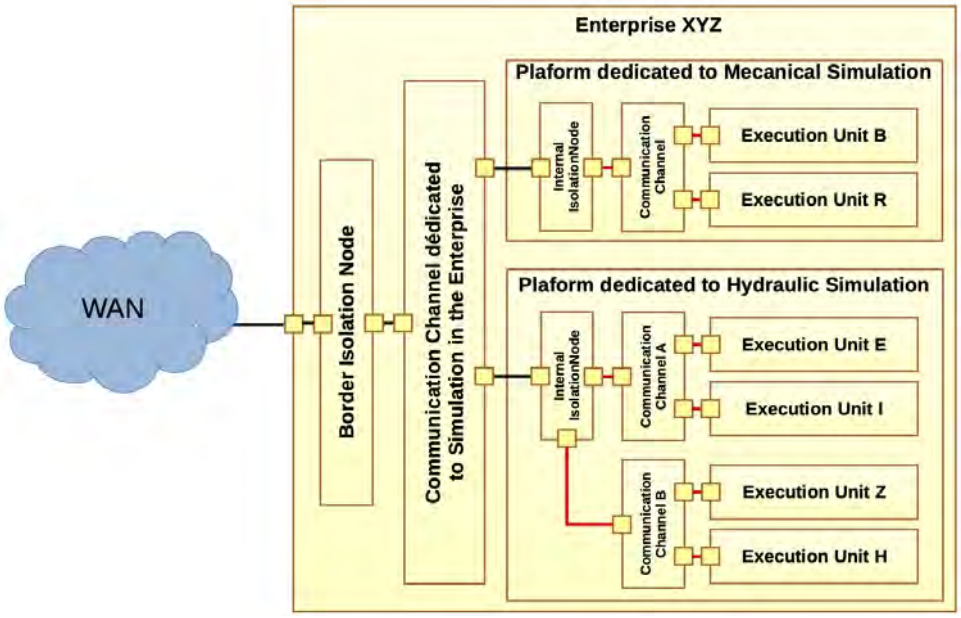

FIGURE 3.15 – Exemple d'une agrégation hypothétique de plateformes de simulation.

Les deux concepts `AdministrationService` et `SimulationService` sont intéressants, dans la mesure où ils permettent de typer les fonctions issues de la phase d'analyse fonctionnelle de la plateforme de cosimulation. Pour illustrer le propos, les fonctions de gestion des droits d'accès, de gestion de la documentation et des modèles peuvent être typées comme `AdministrationService`. Il en est de même principe pour le type `Simulation-Service` auquel les fonctions telles que la gestion de la simulation interactive, la mise à disposition d'un cadriciel de développement logiciel peuvent s'y rattacher.

Maintenant que les concepts sont précisés et structurés entre eux, il devient possible d'instancier ce métamodèle pour modéliser la plateforme de cosimulation en entreprise étendue.

L'instanciation de la méthode générique (section 3.2) et de la plateforme (section 3.3) est décrite dans la section 3.4.

3.4 Instanciation de la plateforme et de la méthode

Après avoir présenté un certain nombre de contributions concernant la méthode générique pour passer du modèle du produit vers les modèles de simulation et leurs exécutions, puis la conception de la plateforme de cosimulation en entreprise étendue, ce chapitre présente une instanciation de cette plateforme de cosimulation (section 3.4.1) et de la méthode

générique associée (section 3.4.2). Ici, le but de l'approche est de mettre en évidence des difficultés de réalisation potentielles, d'y apporter des réponses, des améliorations lorsque cela est possible, et de montrer l'intérêt de la simulation pour chacune des couches système du produit. Ces instanciations sont le passage obligé avant de pouvoir valider (chapitre 4) les propositions présentées.

3.4.1 Instantiation de la plateforme de cosimulation

L'instanciation du modèle de l'architecture physique de la plateforme de cosimulation s'appuie, à la fois, sur l'abstraction par métamodélisation développée à la section 3.3.5.2 (figure 3.14) pour les aspects structurels, et sur l'abstraction par généralisation développée à la section 3.3.5.1, représentée par la figure 3.11, pour les aspects fonctionnels.

Le concept d'entreprise est instancié, à partir du métamodèle, à trois reprises (figure 3.16) pour les modèles d'entreprises `A`, `B` et `Z` (en jaune & liseré marron). Pour illustrer une diversité, parmi tant d'autres, l'entreprise `A` possède deux unités d'exécution, tandis que les entreprises `B` et `Z` n'en ont qu'une seule. Chaque modèle d'entreprise reçoit l'instanciation des concepts de `Communication Channel` et de `Isolation Node`, avec l'ensemble des liens de communications associés, tant en interne que vers le monde extérieur, représenté par le WAN.

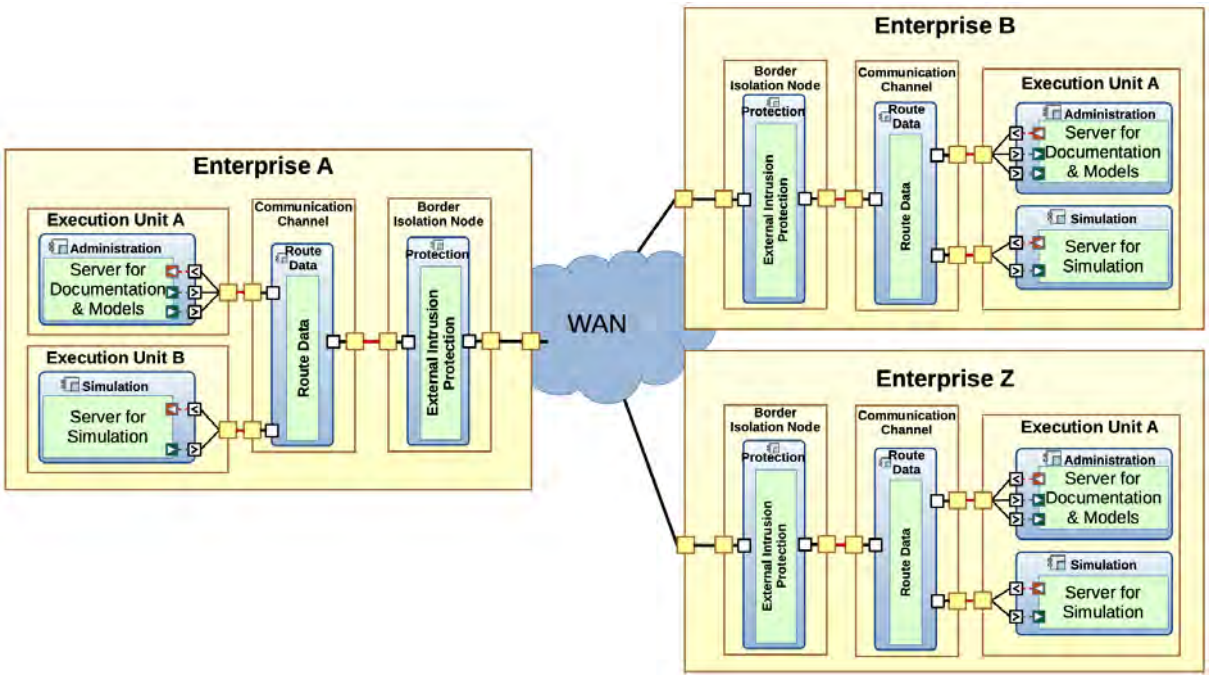

FIGURE 3.16 – Une instanciation de la plateforme de cosimulation.

Cette plateforme de cosimulation fait appel à un ensemble de services. Ces services sont identifiés dans la section 3.3.3, et sont particulièrement représentées par la figure 3.11. Ils servent de patron de conception pour l'ensemble des plateformes de simulation de chaque entreprise.

Chaque fonctionnalité est allouée sur les modèles physiques. Les deux premières fonctionnalités concernent les services de protection contre les intrusions externes (`External Intrusion Protection`) et pour le partage des données dans l'entreprise (`Route Data`), qui sont respectivement alloués aux modèles du composant physique `Border Isolation Node` et `Communication Channel` (les fonctionnalités sont en bleu et vert sur la figure 3.16).

Les deux autres types de services, l'un pour la gestion de la documentation et des modèles de simulation : `Server for Documentation & Models`, l'autre pour l'exécution de la simulation : `Server for Simulation` sont alloués en fonction des choix faits par les entreprises, sur une ou deux unités d'exécution. Dans le métamodèle, ces services correspondent respectivement à `AdministrationService` et `SimulationService`. Par exemple, dans l'entreprise `A` les deux services sont placés sur des unités d'exécution distinctes, alors que pour les entreprises `B` & `Z`, les services sont exécutés sur une même unité.

Sur la figure 3.16, les communications entre les entreprises `A-B` et `A-Z` passent par le réseau internet WAN. Pour ce faire et pour se protéger, les entreprises peuvent adopter une communication de type VPN, de point à point, via l'entremise de leur moyen de protection `Border Isolation Node`.

La mise en place d'une communication de type sécurisée ne semble pas poser de problèmes particuliers entre les entreprises, quelle que soit leur taille. Par contre, il est une difficulté dont la réponse n'est pas forcément immédiate : certaines entreprises peuvent ne pas souhaiter ouvrir un certain nombre de ports de communication supplémentaires, pour accéder de l'extérieur à leurs services internes. Pour elles, cela nécessite de modifier leurs plans de prévention, et potentiellement affaiblir leur protection. Par exemple, les ports dédiés aux protocoles FTP [14], FTPS [15], HTTPS [16] pour le transfert des fichiers ou pour la visualisation de la documentation sont généralement bien surveillés contre les intrusions malveillantes. Qu'en est-il pour l'accès au serveur de simulation, dont le protocole de communication n'est pas forcement standardisé ? C'est un problème potentiel à résoudre pour la réussite de la mise en place de la cosimulation en entreprise étendue.

14. *File Transfert Protocol*
15. *File Transfert Protocol Secure*
16. *HyperText Transfer Protocol Secure*

3.4.2 Instanciation de la méthode générique

La figure 3.17 représente une instance de la méthode générique, représentée par les figures 3.4 et 3.6, et décrites dans la section 3.2.3. Comme pour la figure 3.4, cette figure 3.17 reflète l'indépendance des parties. L'ingénieur du produit choisit par exemple la méthode d'ingénierie système CESAM conformément à son souhait ou à la culture de son entreprise. L'ingénieur système pour la simulation, quant à lui, porte son choix sur la méthode RFLP. Il en est de même pour les développeurs, pour la réalisation de leurs modèles. Ils adoptent leur propre méthode, tout en respectant la technologie de simulation retenue par les partenaires (ici, FMI). Le responsable de l'exécution de la simulation supervise l'intégration des modèles exécutables et prépare la simulation. Il est à noter que, de par le choix de la méthode système basée modèle, l'ingénieur système pour le produit fixe le nombre de lignes de la matrice pour l'ensemble des autres partenaires. Par exemple, dans le cas de la méthode CESAM : 3 lignes, avec la méthode RFLP : 4 lignes.

Les sections suivantes décrivent l'intérêt de la simulation pour chacune des lignes de l'ingénierie système du produit, et les réutilisations possibles entre ces différentes lignes.

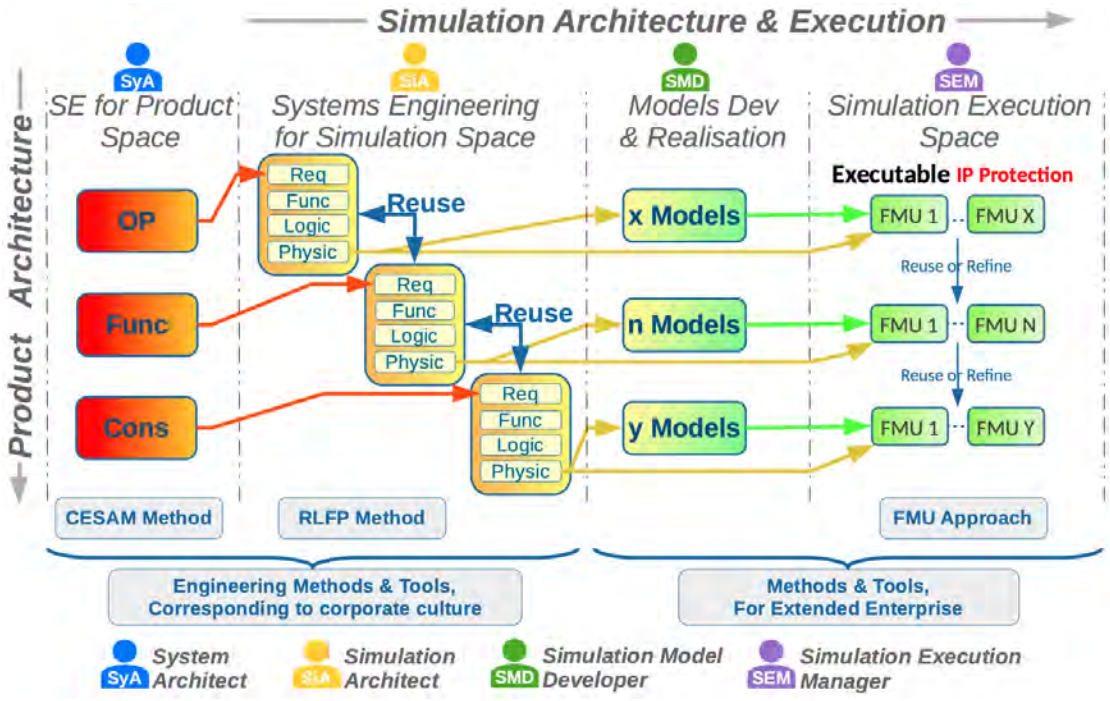

FIGURE 3.17 – Instanciation de la méthode générique.

3.4.2.1 Intérêts de la simulation : étape d'analyse opérationnelle du produit

Pour l'architecte du produit, comme pour l'ensemble des autres acteurs (Architecte système pour la simulation : SiA, les développeurs des modèles de simulation : SMD, les personnes en charge de l'exécution de la simulation : SEM), l'intérêt de la simulation au niveau de l'étape d'analyse opérationnelle du produit est d'initier les scénarios de test servant à vérifier/valider les expressions des besoins dynamiques du client. Le mérite de cette approche est de prendre en compte et de structurer, par des modèles, les échanges avec l'environnement et les acteurs extérieurs au système, tout en définissant la nature des données échangées. Or, à cette étape d'avancement, le système « Produit » est vu comme une boîte noire. Il est donc impératif, pour que ces scénarios puissent être exécutés, de les « bouchonner » avec les réponses attendues.

Le deuxième intérêt est de disposer immédiatement de ces scénarios de test, tout au long de la conception du produit : couche fonctionnelle et couche de construction.

3.4.2.2 Intérêts de la simulation : étape d'analyse fonctionnelle du produit

Lorsque l'ingénieur système du produit SyA aborde la définition de l'architecture fonctionnelle, les scénarios de test de la couche opérationnelle sont, en tout ou partie, déjà fonctionnels et exécutables.

L'architecte du produit, pour ses besoins propres, va chercher à vérifier la pertinence de son architecture, la pertinence de la définition des sémantiques prescriptives de ses fonctions, la cohérence des flux de données échangés. Cette démarche de vérification/validation au plus tôt (« Early validation »), en regard des besoins exprimés par le client, peut être réalisée par la simulation. À cet effet et en fonction d'un objectif de simulation défini, il transmet l'ensemble de ses exigences à l'architecte de simulation. Ces exigences sont listées à la section 3.2.3.6. Une représentation graphique en est donnée par la figure 3.6.

Parmi ces exigences, il en est une qui peut porter à questionnement. Cette exigence concerne la sémantique prescriptive associée à une fonction, particulièrement si cette sémantique est décrite via un modèle de référence exécutable par simulation : Simulink, Modelica, Java, C++ et autres exécutables. La question pourrait être formulée comme suit : pourquoi ne pas utiliser directement au niveau de la simulation, ces modèles exécutables définis par l'architecte du produit ? À cette question, la réponse peut être multiple. Suivant le principe d'indépendance des parties, l'architecte du produit ne développe pas les modèles destinés à la vérification et à la validation. La réalisation des modèles de simulation est une activité dédiée aux développeurs des modèles : SMD.

La réflexion de l'architecte système du produit se fait au niveau système. S'il a bien

une vision d'ensemble du produit et des exigences associées, il ne possède pas forcement les connaissances et le savoir-faire métier du partenaire impliqué dans la réalisation du modèle de simulation réel. À ce niveau fonctionnel, il n'entre pas dans les détails d'implémentation.

Il est possible d'illustrer le propos par un exemple, situé au niveau de la couche système. L'architecte du produit désire que le déplacement du système à l'étude puisse atteindre une vitesse de 25 Km/h en moins de 10 secondes, pour une masse maximale estimée de son système de 5 Kg. Pour ce faire, il fournit une exigence à destination de l'architecte de la simulation, sous la forme d'un contrôleur de type PID [17] (avec les coefficients associés), pour préciser sa pensée. À charge pour le partenaire, qui possède le savoir-faire métier, de créer son modèle de simulation, et de déterminer la puissance de motorisation requise, pour respecter l'exigence tout en prenant en compte les conditions environnementales fournies. Ici, l'intérêt du partenaire est bien de déterminer la puissance de motorisation nécessaire pour répondre à l'exigence et à l'environnement, alors que l'architecte du produit peut ne pas être immédiatement concerné par cet aspect de consommation énergétique induite par la puissance de motorisation mobilisée.

L'autre intérêt de la simulation au niveau de cette étape d'analyse fonctionnelle est de pouvoir réutiliser les fonctions déjà définies (voir la section 3.2.3.6) lors du raffinement de certaines fonctions, tout comme les scénarios de test, issus de l'étape d'analyse opérationnelle. L'architecte système du produit est également amené à définir d'autres scénarios de test de niveau système, lorsqu'une sous-partie du système doit être validée, pour s'assurer que le comportement correspond bien à ses intentions.

Lorsque l'étape d'analyse fonctionnelle est suffisamment raffinée, au niveau de détail désiré, l'architecte système pour le produit peut commencer à travailler sur l'étape d'analyse « Construction ». L'intérêt de la simulation au niveau de cette étape est décrit dans la section suivante.

3.4.2.3 Intérêts de la simulation : étape d'analyse Construction du produit

Au niveau de l'étape d'analyse Construction, l'ingénieur système du produit a déjà son architecture fonctionnelle. Il a pu l'évaluer, la valider, la vérifier par rapport aux spécifications des besoins du client. Dans cette étape d'analyse « Construction », il précise ses choix technologiques et affecte les fonctions de l'étape précédente sur les équipements. Les avantages de la simulation au niveau de cette étape de construction sont quelque peu identiques à ceux évoqués pour l'étape fonctionnelle : réutilisation des fonctions non impactées par les choix technologiques, exécution des scénarios expression des besoins du client, tout comme la réutilisation de certains scénarios spécifiquement développés dans l'étape fonctionnelle.

17. *Proportional Integral Derivative*

3.4.3 Le chaînon manquant entre les modèles et la plateforme

Les deux sections précédentes abordent, l'instanciation de la méthode générique basée MBSE (branche gauche de la figure 1.1) (figure 3.17), et de la plateforme de simulation sur la base de son métamodèle (branche droite de la figure 1.1) (figure 3.14), pour modéliser et réaliser des simulations.

En ingénierie système, la couche physique est destinée à recevoir l'assignation des modèles des composants logiques (dans la méthode RFLP), avec leurs modèles de fonctions associées, sur les modèles des équipements définis dans cette même couche. Dans le cas de l'espace d'ingénierie système pour la simulation (espace de l'acteur SiA, figure 3.17), cette couche physique est quelque peu particulière. Elle représente la convergence de la branche méthodologie (partie gauche, PIM [18]) et la branche des équipements de simulation en entreprise étendue (partie droite, PDM [19]) de la figure 1.1), suivant le concept MDA. Elle reçoit les modèles des équipements physiques de l'architecture physique de simulation de l'entreprise étendue. Cette couche contient, alors, la représentation des calculateurs et autres équipements, dont un exemple en est donné par la figure 3.16.

Avec cette convergence, il devient possible d'assigner les modèles des composants logiques, en provenance de la méthodologie (branche gauche) sur les modèles des calculateurs de la plateforme de simulation en entreprise étendue (branche droite). Par exemple, la figure 3.18 montre l'allocation du modèle de la fonction de simulation `Provide Positioning Signal` sur le modèle du composant de simulation FMU `FMU Position Env`, tout comme pour le modèle de la fonction de simulation `Provide Friction Condition` avec le modèle de composant de simulation FMU `FMU Atmosphere`. Ces deux modèles de composants FMU et le composant `Local Master` sont alloués au modèle de l'unité d'exécution `Execution Unit 1`, située dans l'entreprise `Enterprise XY`.

Toutefois, pour formaliser ces allocations, il est nécessaire d'étendre le métamodèle de la plateforme de cosimulation (figure 3.14) pour lui permettre d'accueillir les concepts de fonction de simulation (`SimulationFunction`) et de composant de simulation (`SimulationComponent`) (figure 3.19).

Le métamodèle de la plateforme de cosimulation (figure 3.14) possède déjà une métaclasse `Allocation`. Cette métaclasse est tout naturellement sollicitée pour y adjoindre les métaclasses `SimulationFunction` et `SimulationComponent` (figure 3.19).

Les flèches pointillées `include` signifient que l'instance de l'élément pointé par la flèche pointillée peut être incluse dans l'instance de l'élément situé à l'origine de cette même flèche. Par exemple, l'instance de l'élément `SimulationFunction` peut être incluse dans l'instance

18. *Platform Independent Model*
19. *Platform Description Model*

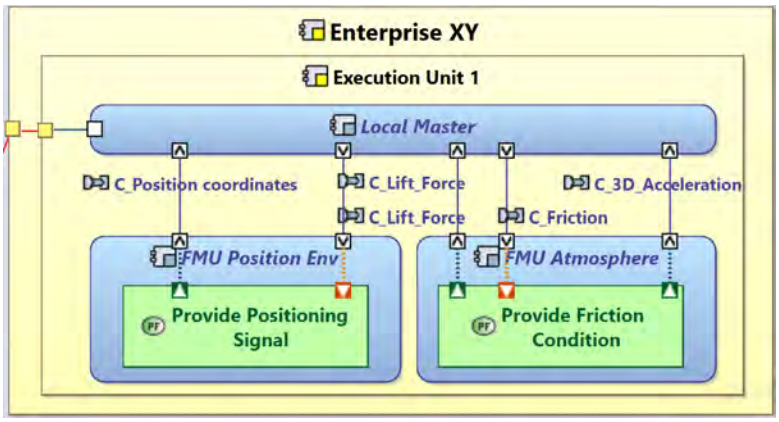

FIGURE 3.18 – Allocation des fonctions et des composants sur les unités d'exécution.

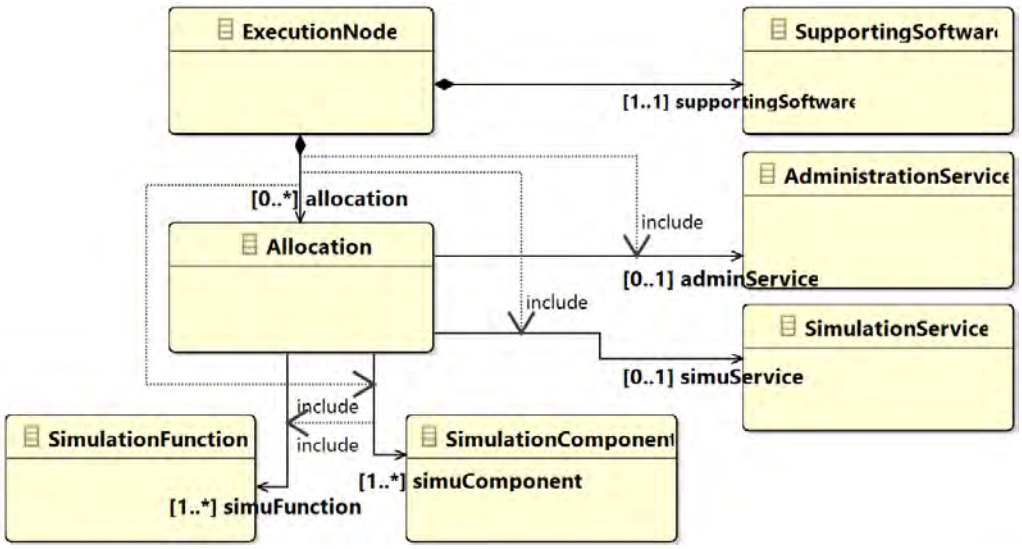

FIGURE 3.19 – Métamodèle de l'allocation des composants sur la plateforme de simulation.

`SimulationComponent`, mais ne peut être incluse dans l'instance de l'unité d'exécution `ExecutionNode`. Par contre, l'instance de `SimulationComponent` peut parfaitement être incluse dans l'instance de `ExecutionNode`(inclusion de type composition structurelle).

Avec cet ajout des liens d'allocation entre les fonctions et les composants, entre ces mêmes composants et l'unité d'exécution, la méthode et la plateforme de cosimulation forment maintenant un ensemble homogène.

3.4.4 L'algorithme « Local Master »

Le standard FMI 2.0 propose un algorithme générique, dit « Master », pour ordonnancer l'exécution et les échanges de données entre les différents FMU, constituant l'ensemble du modèle exécutable de la simulation. Cet aspect générique peut être une limitation dans le cas de la présence de boucles de contre-réaction rapide (ex : 1mS) dans une simulation pouvant être plus lente (ex : 500mS), ou encore dans le cas de la gestion du « rollback », suite au déclenchement d'un évènement entre deux pas de calcul [Ler+17]. Pour ces deux exemples et avec cet algorithme générique, cela nécessite de simuler à la vitesse la plus rapide, ou bien, de faire du « rollback », sur l'ensemble du modèle exécutable de simulation. Outre la perte de temps, il est possible que cette simulation à pas de calcul rapide introduise des effets de bord, tel qu'une instabilité numérique, due à la propagation potentielle d'erreurs d'arrondie.

Si l'organisation de la simulation peut être perçue comme une structure arborescente contenant des circuits orientés (boucles), l'idée de l'algorithme « Local Master » de la figure 3.18 est de prendre en compte ces difficultés, énoncées plus haut, en plaçant judicieusement ces algorithmes « Local Master » sur un certain nombre de nœuds de cette structure arborescente, pour en isoler les boucles et la propagation du « rollback ». Outre les nœuds de simulation, cet algorithme « Local Master » peut être vraiment utile, pour un partenaire qui souhaite développer et cadencer ses simulations derrière l'interface FMI proposée par l'architecte de la simulation.

Pour ce faire, l'algorithme « Local Master » possède deux faces, l'une externe, l'autre interne. L'interface externe est visible du reste de l'arborescence et est de type FMI. L'autre, interne, gouverne le cadencement d'une boucle de simulation, ou gère le « rollback » sur les branches appartenant à ce nœud. De cette manière, vu de l'extérieur, le pas de simulation peut être plus lent (ex : 500mS), tandis que le pas de simulation, interne, de la boucle peut être plus rapide (ex : 1mS).

Maintenant que les modèles sont assignés sur les calculateurs de l'entreprise étendue, il devient aisé de générer l'algorithme « Master », et les API FMU à destination des différentes entreprises. À charge pour elles, et si besoin, de créer leur algorithme de cadencement propre, dans le respect de l'API FMU fournie. Ces générations sont évoquées et non instrumentées dans cette thèse.

3.4.5 Conclusion

Cette section montre qu'il est possible de modéliser l'architecture de simulation de la plateforme de cosimulation tout en respectant le choix de chaque partenaire quant à la

diversité de la mise à disposition d'unités d'exécution nécessaires et suffisantes pour la réalisation de la simulation.

L'instanciation de la méthode générique met bien en évidence le premier principe fondateur de la méthode, à savoir l'indépendance des parties. L'ingénieur système pour le produit utilise sa propre méthode d'ingénierie système, basée modèle, indépendamment de celle de l'architecte de la simulation. Les développeurs, quant à eux, peuvent mettre en œuvre leurs propres approches, tout en respectant le choix fait par le consortium pour la technologie de simulation. De par leurs activités, les personnes en charge de l'exécution de la simulation sont de facto indépendantes. Bien évidemment, les différents outils utilisés par chacune des parties doivent pouvoir communiquer entre eux.

L'ajout, des métaclasses pour les composants et les fonctions de simulation, réalisé sur le métamodèle de l'entreprise étendue, permet de faire le lien entre les modèles exécutables des fonctions et les supports d'exécution, chez chaque partenaire. Le tout forme maintenant un ensemble cohérent, dédié à la simulation.

De plus, sur la base de cette instanciation, cette section souligne l'intérêt de faire de la simulation pour chacune des couches système. La section suivante porte une attention sur l'intérêt de la vérification et de la validation de la plateforme de cosimulation, du simulateur et du produit.

3.5 V&V de la plateforme, du simulateur et du produit

3.5.1 Introduction

Cette section aborde trois types d'activités de validation et de vérification (V&V). La première activité concerne la vérification et la validation en regard de l'expression des besoins de la part des partenaires de cette plateforme de cosimulation en entreprise étendue (sous-section 3.5.2). Lorsque cette activité est réalisée, il devient possible de passer à la V&V unitaire des modèles exécutables de simulation, et donc in fine du simulateur, sur cette même plateforme de cosimulation (sous-section 3.5.3). La troisième activité de V&V, qui se base sur les modèles de simulation considérés comme validés et vérifiés, permet de s'assurer de la pertinence de l'architecture système du produit, en regard des exigences du besoin du client (sous-section 3.5.4). C'est la validation du modèle du produit.

3.5.2 V&V de la plateforme de cosimulation

La section 3.3 a abordé la conception de la plateforme de cosimulation sous l'angle d'une démarche d'ingénierie système basée modèles. Comme tout projet, il devrait faire l'objet d'une V&V sur la base des exigences initiales correspondant à la formalisation des besoins des partenaires, et sur les exigences fonctionnelles et physiques qui en découlent. Ici, il s'agit de s'assurer que l'ensemble des fonctionnalités, matérielles et logicielles, dédiées à cette plateforme de cosimulation sont bien en place et fonctionnelles. Cela concerne, entre autres, la sécurité, la sûreté, les communications entre les plateformes des partenaires, les serveurs dédiés à la documentation et à la simulation. La réalisation de ces tests d'intégration, des activités de vérification et validation est faite par l'IPM de chaque entreprise partenaire. Dans le cas d'une plateforme dont le management est de type hiérarchisé, la supervision de ces activités est réalisée par le responsable IPM.

La vérification et la validation de cette plateforme de cosimulation passe par les tests unitaires de chaque plateforme, chez chaque partenaire, puis par l'intégration d'ensemble, pour vérifier et valider la plateforme de cosimulation. De par la diversité des possibilités, au niveau des locaux, de la sécurité et la sûreté, de l'architecture, des communications et moyens d'exécution, et autres, cette démarche sort du cadre de cette thèse. Pour la suite du propos, cette plateforme de cosimulation est considérée comme vérifiée et validée.

Après l'intégration et la vérification/validation de cette plateforme de cosimulation, le support matériel pour vérifier et valider chaque modèle de simulation devient disponible. La vérification et la validation des modèles des fonctions et de leurs exécutions, la V&V du modèle et de l'architecture du produit peuvent être maintenant abordées.

3.5.3 V&V du simulateur

3.5.3.1 Introduction

Chaque demande de V&V par la simulation conduit à la création, in fine, d'un simulateur demandant des puissances de calcul plus ou moins importantes, avec ses contraintes de réalisation et de coûts. En conséquence, pour maîtriser les contraintes, à l'instar des projets classiques, et pour chaque demande de simulation (figure 3.20, Requirement & Models Product), il est souhaitable de vérifier et valider le simulateur, suivant le modèle du cycle en V classique. Ces activités de V&V concernent le modèle de simulation, constitué de l'ensemble des fonctions afférentes et leurs exécutions sur la plateforme de simulation en entreprise étendue. Cette vérification et/ou validation d'ensemble se fait en deux temps : la vérification et la validation de chaque fonction (figure 3.20, Physic & Soft Models Unary Tests) et leur intégration pour former le simulateur (figure 3.20, Physic & Soft Models Inte-

gration). En dernier ressort, lorsque le simulateur est validé, il sera alors possible d'exécuter les scénarios de test, expression du besoin du client, pour s'assurer que l'architecture du produit, définie par l'architecte système du produit, réponde bien aux exigences du client. Chaque phase pose des problèmes spécifiques. Ces problèmes sont décrits dans les sections suivantes.

3.5.3.2 V&V des fonctions de simulation

Quel que soit le niveau système du produit : opérationnel, fonctionnel ou physique, sur la base des scénarios de test associés aux exigences et objectifs de simulation, le tout remis par l'architecte du produit (SyA), l'architecte de la simulation (SiA) créé son architecture de simulation, réparti les fonctions chez les partenaires sur la base de leur savoir-faire métier, envoie ses propres exigences et scénarios de test, en complément du SyA, vers les développeurs (SMD), ses exigences de placement des fonctions et d'exécution vers les personnes en charge de l'exécution de la simulation (SEM) (figure 3.6). Conformément à la figure 3.20, et chez chaque partenaire, les développeurs réalisent les modèles exécutables de simulation, puis exécutent les tests unitaires adaptés à leurs technologies de simulation, indépendamment les uns des autres. Ces tests unitaires proviennent de leurs propres initiatives, tout comme de la part des mandants que sont les architectes du produit et de la simulation. Pourtant, la décision de statuer sur la recevabilité d'une fonction peut être difficile, en regard des spécifications pleinement, partiellement ou non fournies, par l'architecte système du produit. Ces spécifications partielles peuvent concerner les scénarios de test, la nature et la sémantique de chaque fonction, les propriétés des données d'entrée, de sortie, les oracles de décision, les pré/post conditions, les invariants, les contraintes temporelles, par exemple.

L'architecte système du produit doit fournir obligatoirement, pour chaque fonction, une sémantique associée : explicite ou implicite. Cette sémantique peut être également totale ou partielle, pour un niveau d'abstraction donné. La sémantique explicite décrit ce que la fonction doit faire, c'est-à-dire son mode opératoire. Par exemple, elle peut être décrite totalement ou partiellement par une machine à état, une fonction analytique, un régulateur de type PID ou encore par un programme. À l'inverse, la sémantique implicite précise le but à atteindre sans en indiquer le moyen. Elle fait appel à la compréhension et aux connaissances de la personne en charge de la réalisation du produit. Par exemple, déterminer la racine carrée d'un nombre, ou encore déterminer le frottement entre une roue métallique et la bande de roulement en métal.

Il peut arriver que l'architecte du produit soit dans l'impossibilité d'honorer ces fournitures. Par exemple, il peut être dans l'incapacité de spécifier la loi de distribution d'un signal aléatoire pour l'entrée d'une fonction, ou de fournir un échantillon représentatif des

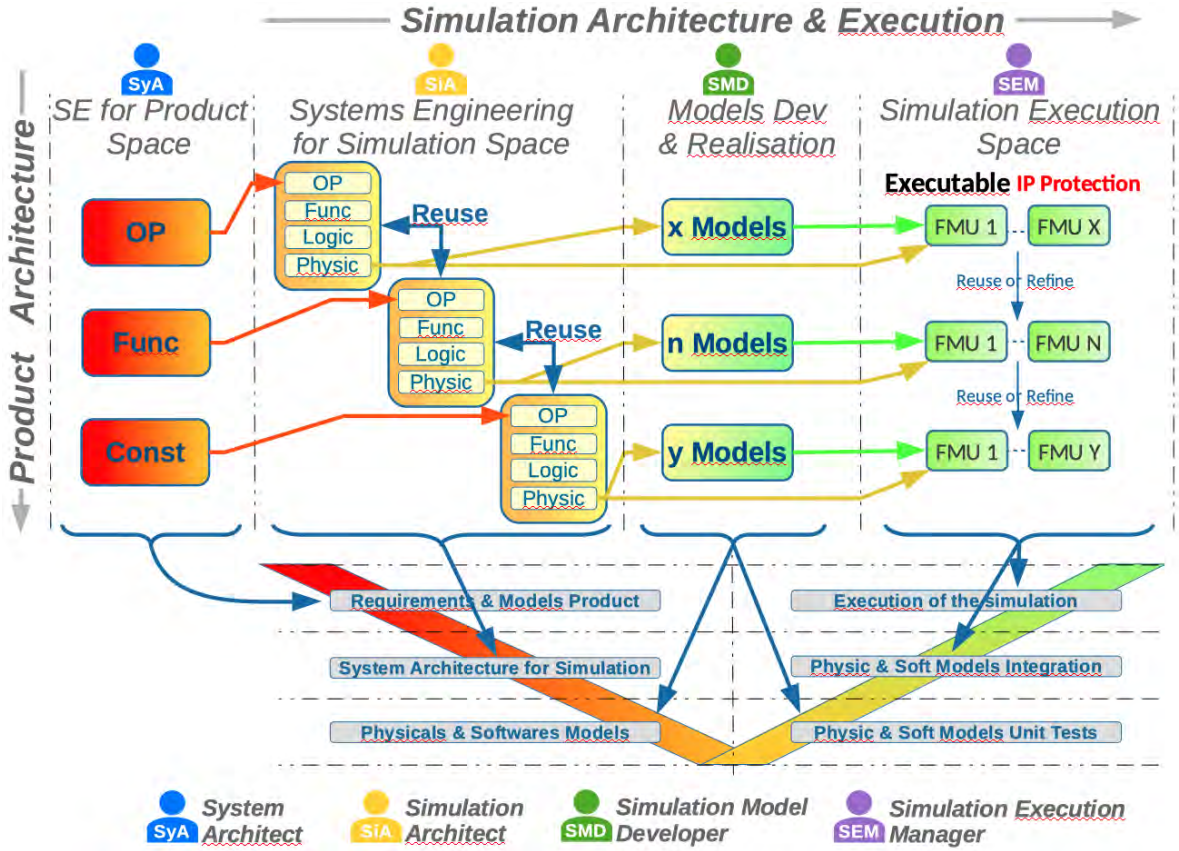

FIGURE 3.20 – Cycle en V des modèles de simulation.

données pour valider une autre fonction, ou encore de préciser un oracle de décision. Dans ce cas, l'architecte système de la simulation (SiA) prend le relais pour essayer de compléter les informations manquantes à destination de la vérification et validation. Pour ce faire, il peut s'appuyer sur des produits existants, des modèles de systèmes précédemment étudiés, sur des standards, ou encore faire appel aux experts des différents corps de métiers pour définir ces données, pour spécifier les scénarios des tests unitaires, ou pour déterminer les oracles de décision. Malgré cette approche, il peut rester des cas où une fonction ne possède pas les éléments suffisants, pour être vérifiée. Pour répondre à cette obligation, Sargent [Sar11] propose d'autres approches. Parmi celles-ci, il y a la comparaison avec d'autres modèles, par exemple avec des modèles proches, analytiques ou non ; des tests de conditions extrêmes, avec des valeurs et paramètres internes limites ; la validité apparente qui consiste à demander aux personnes expertes du domaine, si la fonction à un comportement raisonnable ; ou encore l'emploi de traces d'exécution.

Lorsque l'ensemble des fonctions de simulation sont vérifiées et/ou validées unitairement, il est alors possible de commencer à les intégrer pour valider le modèle de simulation dans son ensemble. C'est l'objet de la section suivante.

3.5.3.3 V&V du simulateur

Dans la section précédente, les tests unitaires se déroulent chez chaque partenaire, indépendamment les uns des autres. Ici, l'intégration des fonctions, entre elles, va donc nécessiter la collaboration de ces mêmes partenaires. Cette phase d'intégration est d'autant plus importante qu'elle peut faire émerger des incompatibilités entre les caractéristiques des données de sorties et des données d'entrées des fonctions, quelle que soit la technologie choisie par les partenaires : FMI, GEMOC, HLA, Ptolemy, et autres. Il est également important de vérifier que les modèles se répondent bien l'un à l'autre (par exemple, dans un mode Push/Pull), de vérifier que le rythme d'échange des données soit compatible des émetteurs et récepteurs, de vérifier qu'il n'y a pas de biais de simulation introduit par la répartition des fonctions chez les partenaires, particulièrement sur les retards dans les boucles de régulation. Dans ce dernier cas, il appartient à l'architecte de la simulation de revoir sa répartition ou de modifier les spécifications de l'algorithme de cadencement.

Nonobstant ces problématiques, l'architecte système du produit (SyA), tout comme l'architecte de la simulation (SiA) fournissent les scénarios d'intégration. Les décisions se prennent selon le verdict des oracles de décision, ou selon les démarches proposées par Sargent [Sar11].

Au final, de proche en proche, les fonctions de simulation sont intégrées entre elles sur la plateforme de cosimulation. Cette démarche aboutit à valider le modèle de simulation dans son ensemble. Plus précisément, c'est bien le simulateur qui est vérifié/validé ; le modèle de simulation et la plateforme d'exécution constituent le simulateur (chapitre 3). Il est crucial que le simulateur soit vérifié/validé. Il devient la fondation nécessaire et sûre, pour vérifier et valider l'architecture du produit, à travers l'exécution des scénarios de test, expression de besoin du client. Il semble important de préciser que la validation du simulateur ne veut pas dire que ce simulateur répond parfaitement aux exigences du client. Cela signifie que le simulateur réagit pleinement aux commandes qui lui sont envoyées.

3.5.4 V&V du modèle du produit

À cette étape, la réalisation du simulateur est terminée. Sa création a été supervisée, vérifiée et/ou validée par l'architecte de la simulation et validée en relation directe avec l'architecte système du produit, son client. Autrement dit, normalement, il ne subsiste plus de doute raisonnable quant à la complétude de ce simulateur et à sa capacité à vérifier/valider les modèles de l'architecture du système. C'est maintenant, sur cette base saine et fonctionnelle, que les scénarios opérationnels, expression du besoin du client, peuvent être exécutés. Lorsque les scénarios sont exécutés, deux cas peuvent se produire. Le premier montre que le modèle répond bien aux attentes exprimées par la spécification de besoin.

L'architecte du produit peut être conforté dans son choix. Lorsque les résultats attendus ne sont pas conformes, dans ce cas, pas de doute, c'est bien le modèle du système qui ne tient pas les spécifications du client, puisque les scénarios de tests et le simulateur ont été pleinement vérifiés et validés en amont. Dans ce cas, l'architecte du produit (SyA) doit revoir son architecture ou bien les paramètres de certaines fonctions. Bien évidemment, ces deux corrections ne sont pas exclusives l'une de l'autre. Ainsi, quel que soit le résultat final, le but initial assigné à la simulation est atteint : montrer que l'évaluation de l'architecture du produit répond ou pas au besoin du client, et ce, bien en amont de la réalisation du produit.

3.6 Contributions des travaux de l'équipe de simulation MOISE

3.6.1 Introduction

Les travaux de la thèse et de l'équipe de simulation MOISE se sont déroulés en parallèle, et avec une certaine forme de séparation, pour laisser à chaque partie la liberté de développer son approche. Ainsi, vers la fin du projet, chacun a pu enrichir l'autre de ses retours d'expériences.

La première version du simulateur, de l'équipe de simulation MOISE, a pour but de mettre en place une chaîne de simulation, sur la base du cas d'utilisation AIDA. Les différentes fonctions ont été modélisées, puis les FMU générés, avec le logiciel SimulationX [ESI19]. L'exécution des FMU et la prise en compte de la notion d'entreprise étendue sont réalisées via le logiciel Cosimate [CHI18]. Les deux entreprises, détentrices des droits sur chacun de leur logiciel, sont membres du projet MOISE. En plus de la mise en place de cette chaîne de simulation, cette première version du simulateur explore la qualité de la simulation, en regard de la fragmentation plus ou moins importante, du modèle de référence, en sous-modèles répartis chez les différents partenaires [Bos+18].

La deuxième version de ce simulateur prend en compte certains aspects de la méthode, notamment la proposition de génération automatique des données pour le squelette du FMU et les interconnexions avec le logiciel Cosimate, l'allocation des fonctions à des composants et aux unités d'exécution, le tout, au sein du logiciel Capella [CAP19]. Le développement, puis la réutilisation, dans son ensemble, du modèle système AIDA (couche Step Physical) oblige à devoir développer des fonctionnalités supplémentaires, telles que la sélection des fonctions, des ports, des flux, uniquement nécessaires à l'objectif de la simulation. Ces fonctionnalités supplémentaires, intégrées dans le méta-modèle de Capella ViewPoint, sont incluses dans cette version 2 du simulateur. Elles offrent une souplesse accrue pour

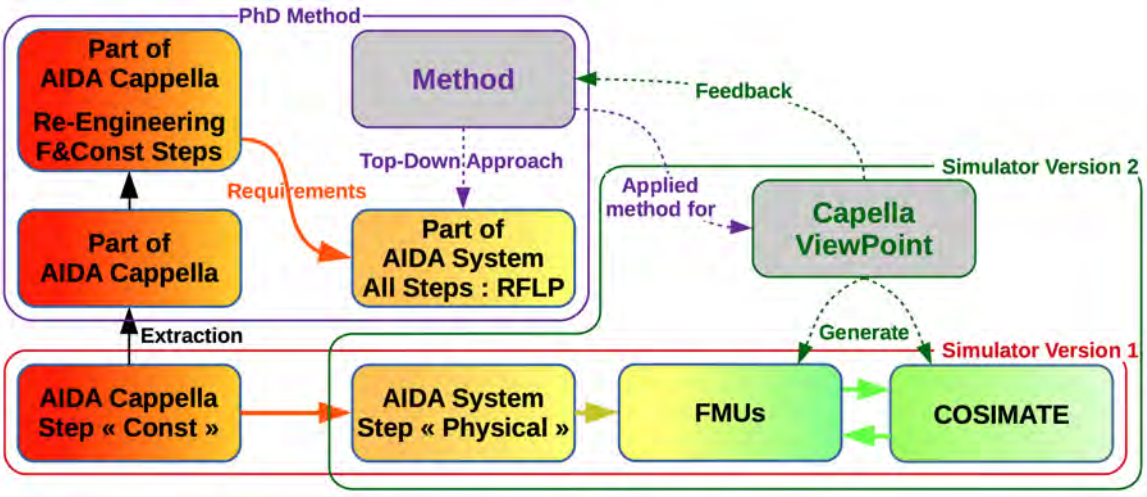

FIGURE 3.21 – Interaction avec l'équipe Simulation-MOISE

bien répondre aux objectifs de simulation.

3.6.2 Les contributions de l'équipe MOISE

La méthode proposée dans cette thèse est cohérente, au sens où elle prend en compte la partie méthodologie, sans oublier le support d'exécution matériel et logiciel constitué par la plateforme de cosimulation. L'architecte système pour le produit fournit la partie du modèle qui doit être simulé, avec les sémantiques prescriptives et descriptives explicites, l'objectif de simulation, et les autres éléments décrits à la sous-section 3.2.3.6. À charge pour l'architecte de la simulation de créer son architecture de simulation.

Cette approche présuppose que l'architecte système pour le produit maîtrise suffisamment bien les domaines techniques entrant dans la conception du produit, que les règles initiales soient bien prises en compte, tant par l'architecte du produit, que par la culture de l'entreprise. Or, lorsque, par manque de connaissances, par une pression potentielle exercée sur l'architecte système du produit, par le non respect des règles, il est possible, devant ces difficultés et pour répondre plus rapidement aux impératifs de l'organisation, que celui-ci fournisse des sémantiques implicites ou encore le modèle dans son ensemble à l'architecte de la simulation. À charge pour l'architecte de la simulation, s'il ne peut faire autrement, de prendre l'ensemble et d'agir en conséquence.

De par sa démarche expérimentale initiale, l'équipe MOISE pour la simulation a transféré l'ensemble du modèle système de l'étape Construction du produit à l'architecte de la simulation. Sur cette base et en fonction des objectifs de simulation, cette équipe a dû développer des concepts et outils spécifiques, sous le logiciel de modélisation système Capella

[CAP19], pour arriver à soustraire, sélectionner, raffiner, abstraire les fonctions devant entrer dans la simulation. Afin d'éprouver la robustesse de l'architecture du drone, objet de leur expérimentation, les membres de cette équipe ont développé des outils complémentaires pour observer (fig 3.22-f) et/ou pour injecter (fig 3.22-e) des erreurs sur les signaux circulant entre les fonctions. Il est à noter que le retour complet de l'expérience de l'équipe MOISE apparaît dans un document de synthèse, dont l'accès est réservé aux membres du projet MOISE. C'est la raison pour laquelle il n'y a pas, dans cette thèse, de référence bibliographique associé à de document. L'objet de cette section est de décrire plus en détail les autres aspects de leurs retours d'expérience. Ce retour d'expérience est représenté graphiquement par la figure 3.22.

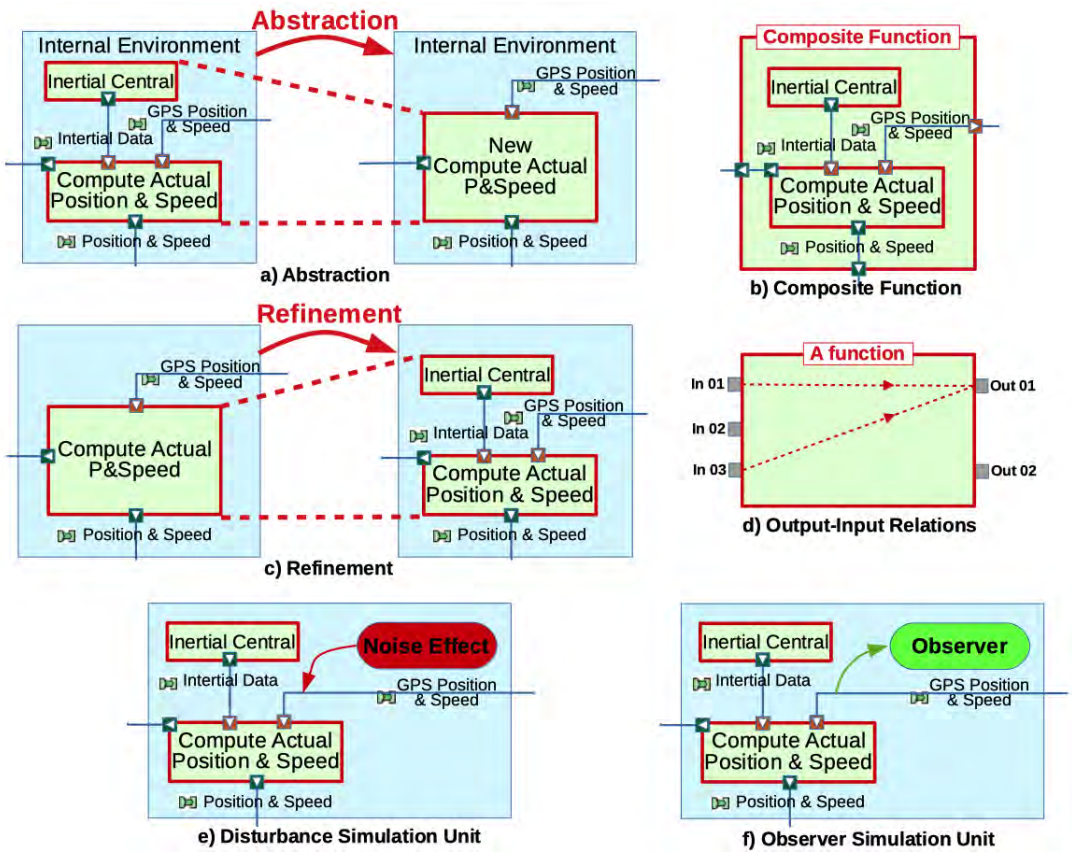

FIGURE 3.22 – Retour Expériences de l'équipe MOISE.

Il est des situations où la mise en œuvre de l'outil Abstraction peut avoir tout son sens. C'est le cas pour des fonctions faisant partie d'un environnement interne (fig 3.22-a). Ces fonctions sont à la fois prescriptive et descriptive. L'architecte de la simulation a, alors, la possibilité de regrouper ces fonctions en ne gardant que les parties spécifiques répondant à l'objectif de la simulation. Il peut en découler un temps d'exécution moindre et une qualité de simulation supérieure, du fait qu'il n'y a qu'un composant de simulation au lieu de deux [Bos+18].

L'architecte de la simulation, pour les besoins de la simulation peut vouloir raffiner une fonction en autant de sous-fonctions désirées, pour, par exemple, répartir les différents constituants chez les partenaires les plus appropriés, par rapport à leur métier. Cette fonctionnalité est représentée par la figure 3.22-c. Cela concerne aussi bien les fonctions dont la sémantique est prescriptive ou descriptive, tout en étant déclarée implicitement ou explicitement.

La Composite function peut être lue de différentes manières. L'une d'elles, c'est de regrouper au sein d'un même composant de simulation (par exemple FMU), des fonctions qui nécessitent un échange de données important. C'est une manière de s'abstraire des temps de transfert entre ces différents composants de simulation. En abordant cette fonction composite sous un autre regard, elle permet de « cacher » derrière une fonction paravent, un ensemble de fonctions qui contiennent le savoir-faire de l'entreprise. La fonction englobante présente l'interface de communication pour la simulation, en conformité avec la technologie de simulation retenue par les partenaires : FMI, HLA, etc. Pour les autres fonctions, le partenaire qui reçoit ce composite a le choix, la liberté d'utiliser la technologie qui lui semble la plus adéquate. Derrière ce paravent que peut être la fonction Composite, il est possible au partenaire d'agréger ses plateformes de simulation, ou encore, d'être en relation avec des partenaires sous-traitants. Enfin, cela semble une manière élégante d'intégrer des phénomènes de couplage mécanique, électrique ou autre, qui ne sont pas initialement pris en compte dans l'architecture du produit, mais qui semble nécessaire d'introduire pour l'objectif de simulation. Suivant les résultats de la simulation, ces couplages peuvent par la suite être intégrés dans l'architecture système du produit, si et seulement si, l'architecte du produit en convient (séparation des responsabilités).

Un autre aspect à prendre en compte lors d'une simulation, ce sont les relations directes (fig 3.22-d et fig 3.23) qui peuvent exister entre les entrées/sorties d'une fonction ("direct feedthrough"), dans laquelle le calcul de la sortie d'une fonction est directement relié à son entrée [Sim19] [FMI14]. Mettre en exergue ces relations directes, permet, via l'analyse des interconnexions entre les fonctions, de déterminer s'il existe des boucles algébriques artificielles ou réelles (fig 3.23). Les boucles algébriques réelles peuvent ralentir ou ne pas permettre de réaliser la simulation demandée de manière correcte.

[Sim19] donne des indications pour savoir si une fonction possède une relation directe entre ses entrées et sorties. La fonction mathématique, la fonction gain, l'espace d'état lorsque la matrice des coefficients n'est pas nulle, la fonction somme, la fonction de transfert lorsque le numérateur et le dénominateur sont du même ordre, ou encore, la fonction Zero-Pole qui possède autant de pôles que de zéros, toutes ces fonctions ont des relations directes entre leurs entrées/sorties. Les fonctions qui ne possèdent pas de relations directes sont les fonctions qui maintiennent des variables d'état. Par exemple, la fonction intégrateur ou la fonction de délais.

Pour illustrer le propos, dans la figure 3.22-d, la sortie Out 01 est directement déterminée avec les entrées In 01 et In 03, tandis que la sortie Out 02 ne dépend pas *directement* des entrées In 01, In 01 ou In 03. Cette dernière sortie dépend de l'état de variables internes à la fonction.

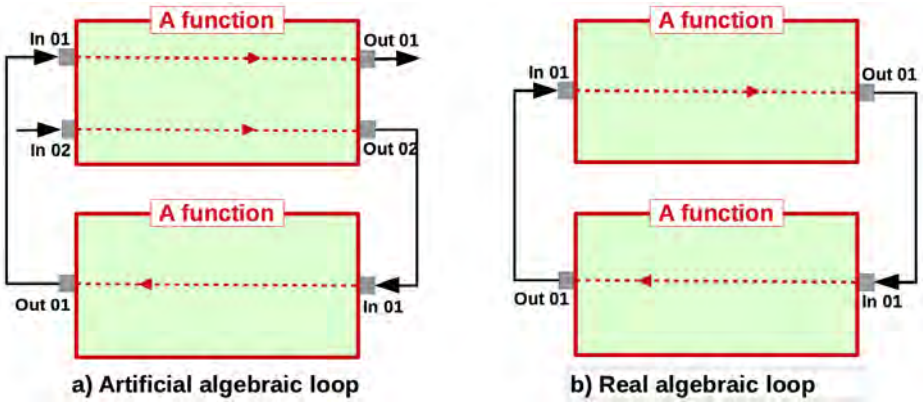

FIGURE 3.23 – Différentes boucles algébriques [FMI14]

L'équipe MOISE a modifié le méta-modèle du logiciel Capella en ajoutant un méta-modèle pour l'espace de simulation, pour mettre à disposition des développeurs ces nouvelles fonctionnalités. Dans le cadre de cette thèse, les ajouts au méta-modèle Capella sont traduits dans le formalisme Ecore [Ste+08], dont le diagramme en est donné par la figure 3.24.

La métaclasse `SimulationElement` représente l'ensemble des éléments du modèle du produit, que l'architecte de la simulation peut recevoir de la part de l'architecte du produit. En se servant de ces éléments et de leurs propriétés, l'architecte de la simulation va pouvoir travailler son architecture, en se basant sur les métaclasses pour abstraire (`AbstractionSimulationUnit`), raffiner (`RefinementSimulationUnit`), observer (`ObserverSimulationUnit`), injecter des erreurs (`DisturbanceSimulationUnit`), créer des composites (`CompositeSimulationUnit`) avec la méta-classe associée (`AtomicSimulationUnit`). Cette extension du méta-modèle Cappela (point de vue) a été implanté dans le cadre du projet MOISE, et a été appliquée pour validation à l'étude du cas AIDA [20].

3.7 Contributions à destination des industriels

La contribution présentée dans cette section va au-delà du cadre de la définition de la méthode de conception d'une plateforme de cosimulation. Elle met en avant des élé-

20. *Aircraft Inspection by Drone Assistant*

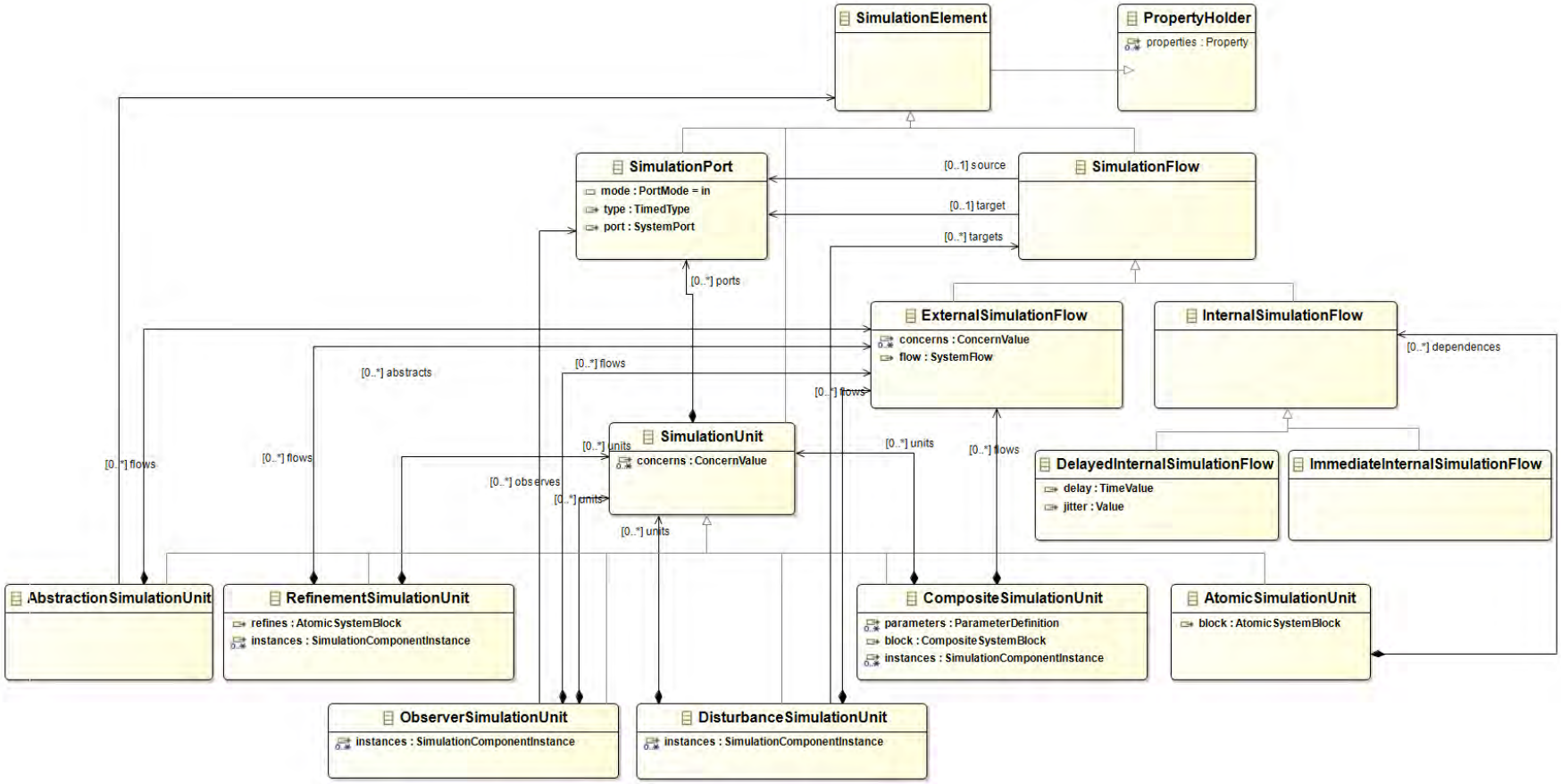

FIGURE 3.24 – Extrait du Méta-modèle MOISE de l'espace de l'architecte de simulation

ments importants pour l'industrialisation. Ces éléments permettent de prendre en compte l'environnement de cette plateforme tant pour les aspects techniques que pour les aspects administratifs, contractuels et de sécurité.

Ces sections ont vocation à lister de manière non exhaustive de potentiels problèmes sous forme de questions. Bien évidemment, il appartiendra aux futurs acteurs de trouver les réponses les plus adéquates, en fonction du sens qu'ils veulent donner à cette entreprise étendue et en fonction de leur culture d'entreprise respective.

3.7.1 Clauses de confidentialité

En premier lieu et suivant la pratique usuelle dans l'industrie, il est souhaitable de signer des clauses de confidentialités entre les partenaires initiateurs de cette entreprise étendue. Cela permettra plus aisément de traiter et contractualiser, en amont du projet, ces deux points essentiels : a) Quelles formes juridique et fonctionnelle l'entreprise étendue doit-elle prendre ? b) Comment gérer la propriété intellectuelle issue des travaux de l'entreprise étendue, ou connexe à ces travaux ?

Il est possible d'avoir une première piste de réflexion, en se basant sur le contenu du paragraphe 2.5 qui liste des organisations de base pour l'entreprise étendue. Toutefois, choisir l'une de ces typologies, ou plus possiblement une composition de celles-ci implique une organisation fonctionnelle et juridique. Or, suivant l'organisation juridique choisie, celle-ci peut contraindre les droits sur la gestion et la répartition de la valeur ajoutée de la propriété intellectuelle, et donc la réponse à la deuxième question. Obtenir la cohérence d'ensemble : fonctionnelle et juridique, oblige certainement à un processus itératif de décisions collégiales entre partenaires, dans un esprit « gagnant-gagnant ».

3.7.2 La propriété intellectuelle, entreprise entrante/sortante

Les acteurs, fondateurs de l'entreprise étendue, comme les futures entreprises entrantes, peuvent apporter leurs savoir-faire sous forme documentaire, logicielle, matérielle, de brevet, ou encore de compétences. Ces apports initiaux souvent appelés « backgroung IP » doivent faire l'objet d'une mention explicite et d'une évaluation de leur valeur, pour déterminer la contribution de chacun. Quels sont les critères à prendre en compte pour évaluer ces contributions ? Pour cette question, une des possibilités est de s'appuyer sur une structure extérieure spécialisée dans la PI [Jea12].

Pour les brevets entrants, au sein de l'EE, un partenaire peut accorder une licence d'exploitation. Sur quelle durée, pour quel coût ? La licence d'exploitation est-elle conta-

minante ? Sa simple utilisation implique-t-elle que tout élément associé à l'objet de cette licence devient la propriété de son auteur ?

Pour une entreprise entrante en cours de projet, quelle part peut-elle prétendre sur la valorisation et l'exploitation de la PI produite, en fin de projet ? La même question se pose pour une entreprise sortante en cours de projet. D'après [Jea12], il existe des outils de type *workflow* qui « permettent de gérer les étapes relatives à la PI en fonction des phases de la collaboration ». C'est une aide possible pour répondre à cette question.

Que se passe-t-il si un partenaire quitte l'entreprise étendue en cours de projet (par exemple, cessation d'activité, ou bien renoncement) ? Que deviennent ses contributions et résultats, vis-à-vis des autres partenaires ? Dans le cadre de cette thèse, que deviennent les modèles de niveau système ? Idem pour les modèles de simulation de ce partenaire ? Plusieurs possibilités :
a) le modèle est un code binaire exécutable. Le partenaire reverse cet exécutable à l'entreprise étendue, à quel coût ? En cas de défaut, faut-il prévoir une clause de maintenance, dès le début du projet ?
b) l'exécution du modèle fait appel à un logiciel développé en interne, chez ce partenaire. Doit-il fournir ce logiciel pour exécuter son modèle, au risque de devoir dévoiler une partie importante de son savoir-faire, et à quel coût ?

Comment se répartir les droits d'exploitation entre partenaires, sur une PI produite en marge de ce même projet, et sur quelle durée ?

Si des résultats ne peuvent faire l'objet d'une PI, peuvent-ils être diffusés à l'extérieur de cette entreprise étendue, sous quelles conditions ?

3.7.3 Exigences centrées sur les servitudes

Les servitudes d'une plateforme de simulation concernent aussi bien les besoins pour les personnes, comme pour les unités d'exécution. Par exemple, dans les locaux et pour les personnes, il faut prévoir, tant pour respecter la loi que pour le confort de celles-ci : une protection incendie, une aération, un éclairage suffisant, une régulation de température, des points d'eau, la téléphonie et toutes autres fonctions nécessaires à la préservation des personnes. Pour l'aspect matériel, les alimentations électriques, des points d'accès réseaux sont indispensables. Les moyens de protection contre les intrusions, suivant le contexte, sont à considérer.

3.8 Conclusion

Ce chapitre a précisé et défini deux grandes lignes pour faire de la simulation, un processus de vérification et validation des fonctionnalités d'un produit.

L'exécution d'une simulation en entreprise étendue s'appuie sur des plateformes de simulation, situées au sein de chaque entreprise. La proposition aborde la définition de cette plateforme de simulation sous l'angle de l'ingénierie système, basée modèle. De cette étude, il en découle une liste d'exigences, la définition d'acteurs et de leurs rôles, une architecture fonctionnelle et physique. Après factorisation des éléments de cette étude, il en ressort une l'architecture fonctionnelle, qui peut être considérée comme architecture de référence pour l'ensemble des plateformes de simulation de l'entreprise étendue. Avec un autre regard, il a été possible de créer un métamodèle qui agrège les concepts issus de l'analyse de la couche fonctionnelle et de la couche physique. Cette dernière abstraction sera utilisée dans la section suivante (section 3.4) lors l'instanciation de la méthode.

La deuxième grande ligne pour faire de la simulation a été de définir une méthode qui prenne en considération à la fois les acteurs, l'ingénierie système du produit comme point de départ pour arriver à la simulation sur la plateforme de cosimulation en entreprise étendue, tout en assurant la traçabilité des exigences. Cette méthode peut s'appliquer à chaque étape de la conception du produit (couches expression du besoin, fonctionnelle, ou encore, la couche physique). Elle est la « pierre angulaire » de cette intégration de la simulation, pour la vérification & validation, au plus tôt de la conception du produit. Elle s'appuie sur les abstractions de la plateforme de simulation, précédemment étudiées.

Le retour d'expérience précieux de l'équipe MOISE pour la simulation, avec la mise en place du point de vue Capella, permet d'avoir une souplesse quant à la sélection des éléments utiles à la simulation. Si la méthode développée dans cette proposition est cohérente, ce retour permet de prendre en compte les diversités de travail, d'expériences des architectes système pour le produit, en offrant la possibilité aux architectes système pour la simulation, de pouvoir ajuster au mieux, leur architecture de simulation.

Enfin, si certains éléments de réflexion, quant à la constitution d'une plateforme de simulation, n'entrent pas directement dans l'objet de cette thèse, il semble utile de les conserver comme possibles contributions pour les industriels. Cela concerne les clauses de confidentialités, la propriété intellectuelle et les exigences centrées sur les servitudes associées aux plateformes de simulation.

Le chapitre 4 suivant aborde la validation de la méthode proposée dans cette thèse, sur le cas concret développé par l'équipe MOISE : le drone AIDA. Plus spécifiquement, cette validation s'appuie sur les fonctionnalités associées au pilotage automatique de ce drone.

Chapitre 4

Validation de la méthode

Sommaire

4.1 Introduction

Ce chapitre aborde la validation de la démarche générique proposée et instanciée dans le chapitre précédent. Cette démarche est éprouvée sur le cas d'étude AIDA, dont la description est donnée à la section suivante.

Pour valider notre proposition, nous considérons la démarche d'un ingénieur système employant la méthode exprimée par la figure 3.20, pour définir l'architecture système de son produit. En premier lieu, l'architecte système pour le produit commence par traiter l'étape opérationnelle, c'est-à-dire par modéliser les exigences relatives aux besoins de son client. Il transmet ces exigences à l'architecte de la simulation, qui utilise sa propre méthode (ici, RFLP) pour dresser l'architecture de simulation. Cette démarche est identique pour les étapes : fonctionnelle et de construction du produit. L'ensemble des diagrammes de ce chapitre sont des copies d'écran des modèles réalisés avec le logiciel « Capella » [CAP19].

4.2 Cas d'étude AIDA

Le cas d'étude AIDA (Aircraft Inspection by Drone Assistant), sélectionné et réalisé par l'équipe projet MOISE, consiste en un drone d'inspection d'avion. Ce cas d'étude est en open source et librement disponible [IRT19a].

Avant tout décollage de l'avion, ce drone réalise un parcours prédéterminé (figure 4.1), pour permettre au pilote de visualiser et détecter les éventuelles irrégularités sur les parties non directement accessibles : dessus des ailes, dessus du fuselage, l'empennage. À cet effet, le drone possède la cartographie de l'avion et les points d'intérêt à scruter. Les irrégularités à détecter peuvent être de plusieurs types : l'oubli des protections sur les senseurs, des trappes ou portes mal fermées, des effets mécaniques tels que des coups de foudre, ou des impacts de grêles.

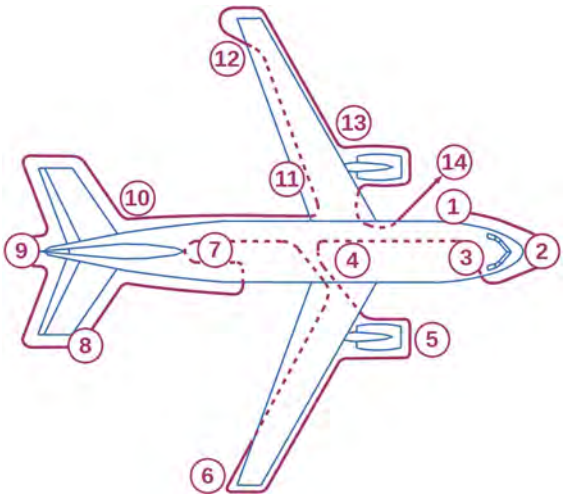

FIGURE 4.1 – Inspection visuelle.

Pour remplir sa mission, le drone est équipé d'une variété de capteurs tels qu'un système de vision, un positionnement GPS, un radar pour une plus grande précision quant à la distance qui doit être maintenue entre celui-ci et l'avion ou les personnels au sol. De plus, le pilote peut visualiser en temps réel les images prises par le drone, pour au besoin, corriger le plan de vol pour une inspection plus approfondie.

En cas de dysfonctionnement du drone et à des fins d'analyse, les données internes et les données du vol sont transférées en temps réel vers le sol, en plus d'être sauvegardées localement.

Dans un souci de simplification et de lisibilité, la mise en œuvre de la méthode se concentre sur le pilotage automatique, l'environnement atmosphérique et la partie localisation du drone.

Pour valider le bien-fondé de la méthode, des objectifs de simulation sont posés. *L'objectif du client est de s'assurer que le drone réalise bien et automatiquement le plan de vol spécifié.* Cette exigence est appliquée aux couches système fonctionnelle et physique. Au niveau de la couche fonctionnelle, il s'agit de vérifier que la fonction de déplacement répond bien aux commandes en provenance du plan automatique d'inspection. Au niveau de la couche physique, il est nécessaire de vérifier que la motorisation avec 4 hélices est suffisante pour respecter l'exécution du plan de vol. À cette dernière exigence, s'ajoute l'opportunité de vérifier que les délais introduits par les bus de communication, au sein du drone, n'introduisent pas des biais qui rendraient la mise en œuvre de ce plan de vol d'inspection difficile.

4.3 Validation de la méthode

4.3.1 Introduction

L'objectif est d'illustrer l'application de la méthode sur un exemple réaliste pour démontrer l'applicabilité et la tenue des exigences. La question se pose : comment valider cette méthode générique, représentée sous différents angles, par les figures 3.4, 3.6, 3.7, 3.8, dont une instanciation est donnée par la figure 3.17, tout en y incluant la plateforme de simulation en entreprise étendue, mise en exergue par les figures 3.11, 3.12, 3.16 ?

À l'instar de Sargent [Sar11], il semble que la validation par les Pairs soit l'approche la plus appropriée. Dans ce chapitre, la validation s'appuie sur le cas d'usage AIDA, issu des travaux du projet MOISE. Ce moyen d'inspection AIDA est la solution vers laquelle l'architecte système du produit doit converger. Toutefois, avant d'arriver à cette solution, l'architecte système du produit commence sa réflexion en s'appuyant, sur la méthodologie MBSE, dite CESAM. La mise en œuvre de cette méthode, via le cas concret AIDA, est déroulée dans les sections suivantes.

Le modèle AIDA développé par l'équipe MOISE contient une bonne centaine de fonctions. Ce modèle AIDA a été réduit à un sous-ensemble de ses fonctionnalités, pour le besoin d'une démonstration simple et facilement lisible de la méthode. Ce sous-ensemble concerne la partie : inspection automatique (figure 4.2, « Manage Inspection Plan » & « Automatic Inspection ») et la fonction de déplacement (figure 4.2, « Compute Movements Consigns » & « Mecanical Move »). Lors de cette demande de simulation, l'architecte du produit fixe l'objectif de la simulation, à savoir : vérifier que la fonction de déplacement est bien appropriée pour exécuter le plan d'inspection.

4.3.2 L'étape opérationnelle

Comme évoqué à la section 3.4.2.1, le but de la simulation au niveau de cette étape opérationnelle est de mettre en place les scénarios de tests de l'expression du besoin du client. Pour l'objectif de la simulation, donné dans le paragraphe précédent, ce scénario de test se résume à l'exécution d'une commande « Start » pour lancer le mobile dans son parcours d'inspection. Ce parcours est préalablement chargé en mémoire sur le mobile.

4.3.3 L'étape fonctionnelle du produit

Dans cette étape d'analyse fonctionnelle du produit, l'architecte du produit cherche à construire, puis à vérifier la pertinence de son architecture et de la sémantique des fonctions associées. Dans cet exemple, fil conducteur de la démonstration, il cherche à vérifier que la fonction déplacement, assurée par « `Compute Movements Consigns` » & « `Mechanical Move` », soit suffisamment réactive, pour exécuter le plan d'inspection, préalablement chargé en mémoire. Le modèle fonctionnel est représenté par la figure 4.2. En bleu clair, ce sont les fonctions d'environnement du modèle. En vert clair, les fonctions du modèle qui sont concernées par l'objectif de la simulation.

Il est à noter que dans cette couche fonctionnelle, l'architecte du produit ne précise pas le type du mobile : nacelle élévatrice, drone, ou bien encore piéton équipé d'une perche avec caméra à son extrémité.

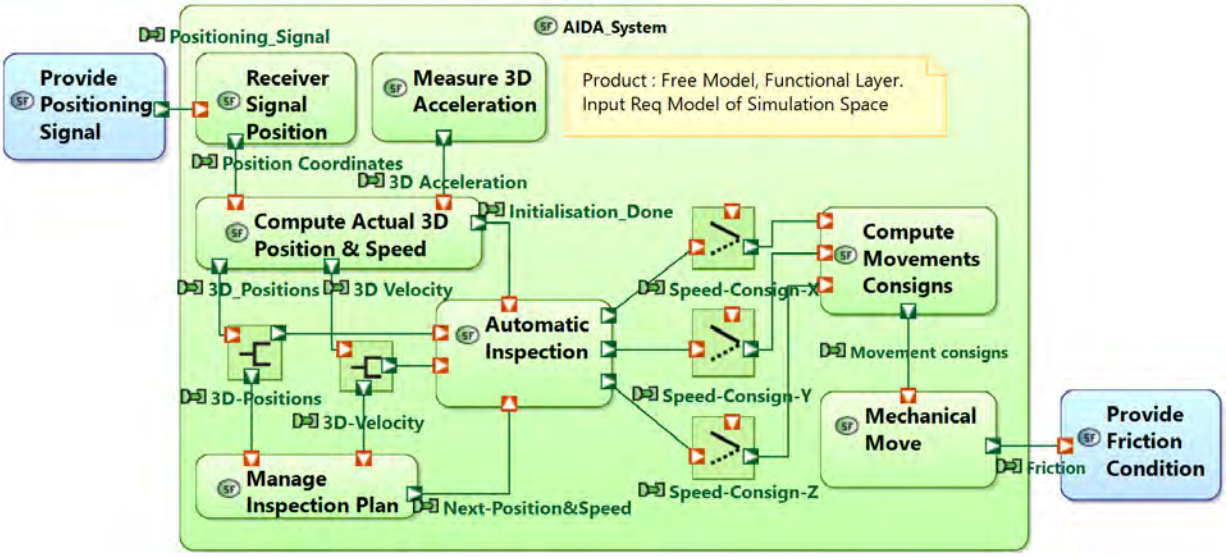

FIGURE 4.2 – Produit : Étape fonctionnelle – Espace de simulation : Exigences d'entrée.

4.3.3.1 Produit : Étape fonctionnelle – Espace de simulation : Exigences d'entrée

L'architecte de la simulation reçoit le modèle représenté par la figure 4.2 en provenance de son client : l'architecte système du produit. Ce modèle sert d'exigence d'entrée pour la simulation. À ce modèle, l'architecte du produit associe, en autres, la description du rôle de chaque fonction. La fonction « Provide Positioning Signal » fournit un signal de positionnement. Le récepteur de position « Receiver Signal Position » traduit le signal reçu, en coordonnées de position à destination de la fonction « Compute Actual 3D Position

& Speed ». Cette fonction corrobore la position déterminée par les deux intégrations successives de l'accélération 3D reçue, avec celle en provenance du récepteur. Les messages « 3D position » et « 3D Velocity » sont diffusés aux fonctions « Manage Inspection Plan » et « Automatic Inspection ». Lorsque le mobile arrive à la position souhaitée, la fonction « Manage Inspection Plan » fournit les coordonnées de la prochaine position à atteindre à destination de la fonction « Automatic Inspection ». Cette fonction compare la position courante du mobile avec la position cible. Elle envoie la correction vers la fonction « Compute Movements Consigns » qui traduit cette information en mouvement à réaliser par le mobile. Enfin, la fonction « Provide Friction Condition » simule les conditions de frottement, pour que le mobile puisse se déplacer.

Ce sont bien ces informations, avec celles décrites dans la section 3.2.3.6, qui sont l'expression des exigences de l'architecte du produit.

4.3.3.2 Produit : Étape fonctionnelle – Espace de simulation : Étape fonctionnelle

Dans cet espace d'ingénierie système pour la simulation et au niveau de l'étape fonctionnelle, l'architecte de la simulation recherche les contraintes pouvant avoir un impact sur la qualité et la durée de la simulation, sans pour autant apporter une plus-value significative à l'objectif de la simulation. Ces contraintes peuvent, par exemple, être des fonctions fortement consommatrices en ressources d'exécution, des débits entre partenaires trop déséquilibrés, des simulations bien trop précises sur des signaux bruités. Dans les exemples donnés, le premier cas peut nécessiter de revoir une partie de l'architecture du modèle d'entrée, pour l'adapter à la simulation (figure 4.3) afin d'améliorer les temps de simulation. Dans le même esprit et pour le deuxième exemple, un rééquilibrage des charges entre partenaires, via une optimisation tenant compte du débit, pourrait permettre d'obtenir les résultats de simulation plus rapidement. Enfin, dans le dernier cas de cet exemple, si une fonction prend en entrée un signal bruité et sa dérivée première, il peut être nécessaire de mettre en place un filtre de lissage, pour limiter l'étendue potentielle des valeurs que pourrait prendre cette dérivée et pour assurer une qualité de simulation bien meilleure.

Avec la figure 4.3 et pour illustrer plus amplement le premier exemple, l'architecte de la simulation détecte que les fonctions « `Receiver signal Position` » et « `Accelerometer` » sont chronophages, tout en ayant un impact faible sur l'objectif de la simulation. C'est la raison pour laquelle, en accord avec l'architecte système du produit, il modifie l'architecture du modèle pour réduire les coûts de simulation. Pour ce faire et sur la base du mouvement généré par la fonction « `Mechanical Move` », il modifie la fonction « `Provide Friction Condition` » pour que celle-ci fournisse l'accélération et le mouvement du mobile. L'information d'accélération est directement connectée à l'entrée de la fonction « `Compute`

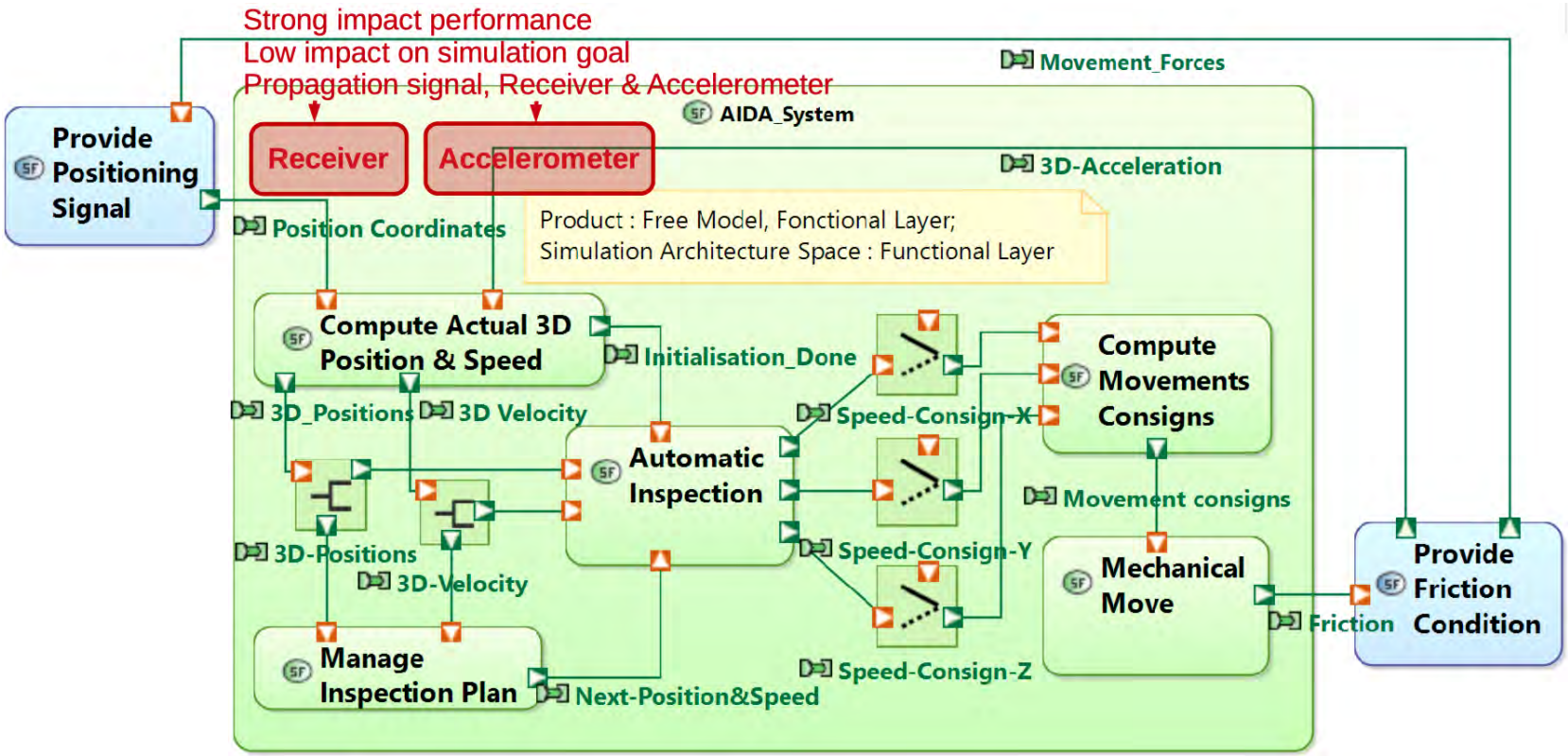

FIGURE 4.3 – Produit : Étape fonctionnelle – Espace de simulation : Étape fonctionnelle

`Actual 3D Position & Speed` » telle qu'attendue, et le mouvement est calculé dans la fonction « `Provide Positioning Signal` » pour fournir une position à l'entrée de la fonction « `Compute Actual 3D Position & Speed` ». En agissant ainsi, l'architecte de la simulation évite de devoir spécifier la modélisation du système produisant le signal de position, de modéliser la propagation d'onde, de modéliser le récepteur avec l'ensemble de ses changements de fréquences intermédiaires, et pour finir de modéliser l'accéléromètre.

Dans l'instanciation de la méthode générique, figure 3.20, l'architecte de la simulation utilise la méthode d'ingénierie système RFLP. Lorsque le travail de simplification potentielle au niveau de cette étape fonctionnelle est réalisé, l'architecte de la simulation poursuit sa démarche au niveau de l'étape logique.

4.3.3.3 Produit : Étape fonctionnelle – Espace de simulation : Étape logique

Dans cette étape logique, l'architecte de la simulation regroupe les fonctions de manière logique. Plus précisément, les fonctions sont rassemblées en regard des partenaires et de leurs spécificités métiers. Cette démarche est illustrée par la figure 4.4, où l'indicateur « `LC_` » signifie « `Logic Component` » et dans lesquels les fonctions sont allouées.

Le raffinement de la fonction « `Mechanical Move` » n'est pas présent dans la figure 4.3. Toutefois, le détail de cette fonction (figure 4.4) est affiché dans le composant logique « `LC_Mechanical_Move` ».

Lorsque l'ensemble des fonctions sont assignées à des composants logiques, l'architecte système pour la simulation peut poursuivre sa démarche au niveau de la couche physique.

4.3.3.4 Produit : Étape fonctionnelle – Espace de simulation : Étape physique

Cette étape physique est particulière, dans la mesure, où elle rassemble les contributions issues de l'étude de la plateforme de simulation et les contributions dues au déroulement de la méthode générique : s'appuyer sur le modèle du produit pour créer les éléments de simulation, et pour enfin, créer les simulateurs.

Les contributions de la plateforme de cosimulation sont, entre autres, de prendre en compte les entreprises et leurs unités d'exécution. Elles apparaissent via les instances des métaclasses, sous les dénominations « `Entreprise A` » .. « `Entreprise Z` », puis les « `Execution Unit 1` » et « `Execution Unit 2` ». Par mesure de simplification et de lisibilité, la figure 4.5 ne représente pas les éléments « `IsolationNode` » et « `Communication Channel` », identifiés au cours de l'analyse système d'une plateforme de cosimulation (section 3.3).

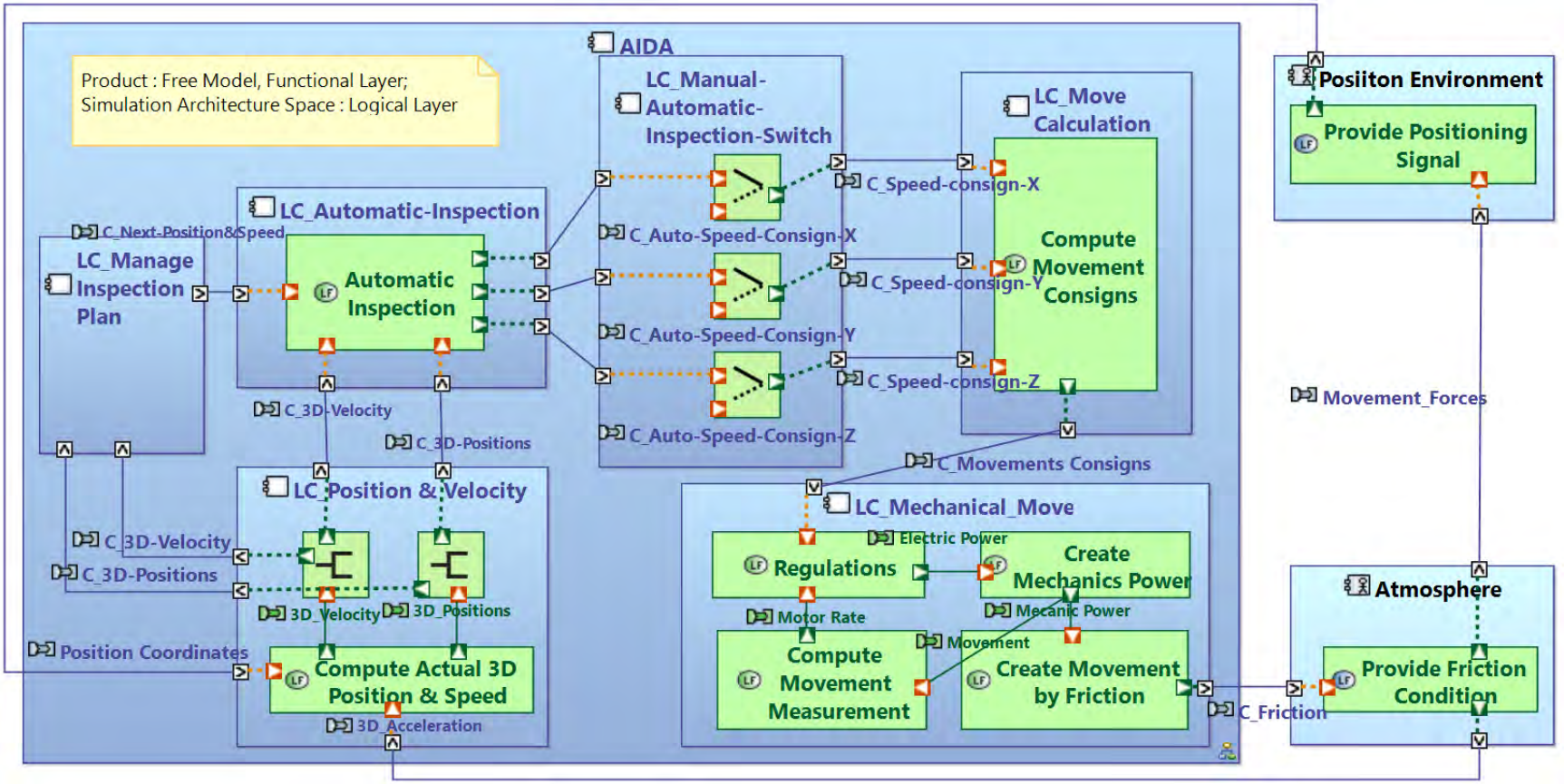

FIGURE 4.4 – Produit : Étape fonctionnelle – Espace de simulation : Étape logique.

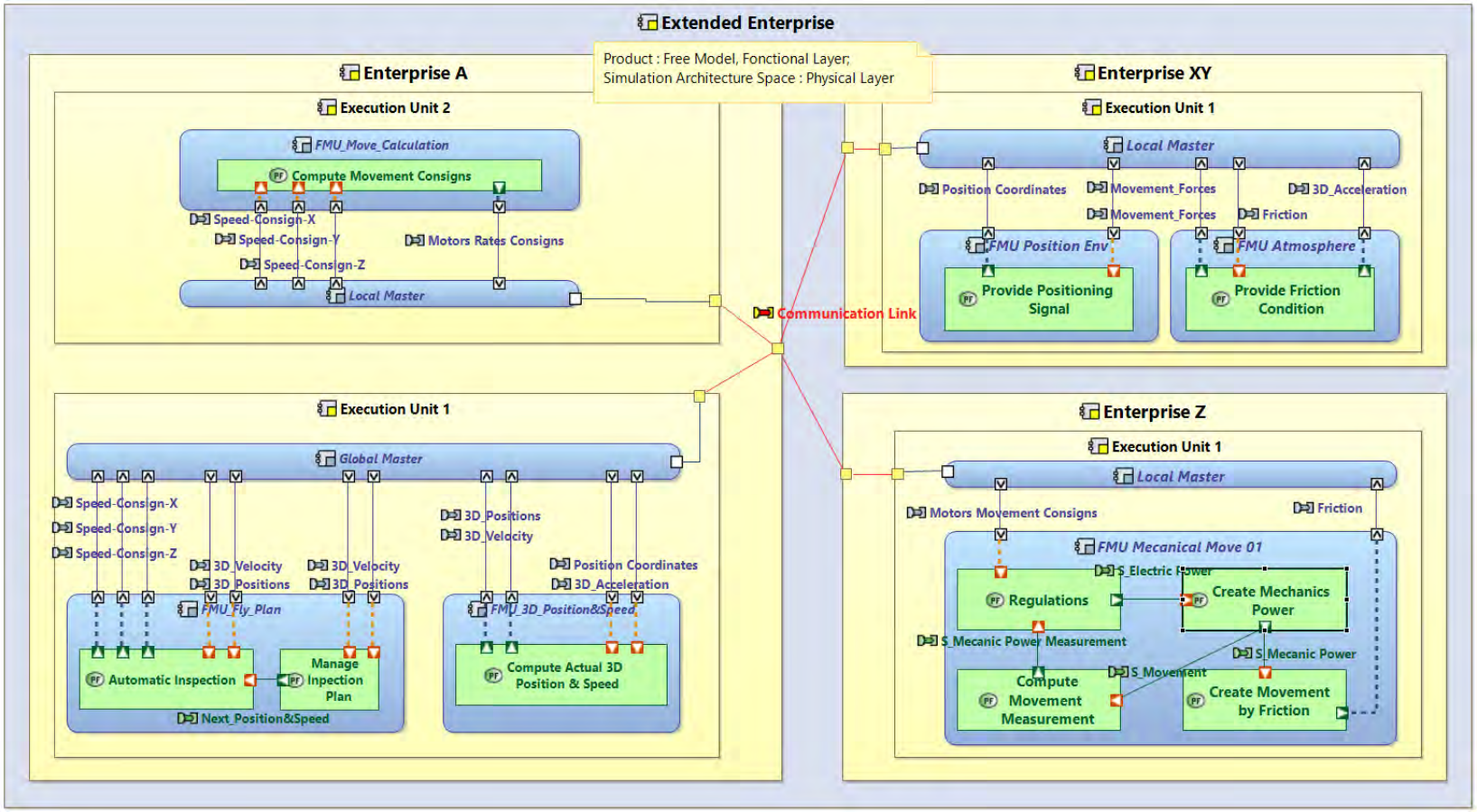

FIGURE 4.5 – Produit : Étape fonctionnelle – Espace de simulation : Étape physique.

Au niveau de l'espace de l'architecte système pour la simulation, l'une des contributions, issue de la méthode, est de pouvoir répartir chez les partenaires, et sur leurs unités d'exécution, les composants de simulation. Par exemple, les composants « `FMU_Fly_Plan` », « `FMU_3D_Position&Speed` » et le composant « `Global Master` » sont répartis sur l'unité d'exécution « `Execution Unit 1` », elle même située au sein de l'entreprise « `Enterprise A` » (figure 4.5).

Cette répartition est rendue possible par la modification du métamodèle de la plateforme de cosimulation (cf section 3.4.3).

La méthode se veut générique. Elle est agnostique quant au choix de la technologie de simulation : FMI [FMI19], HLA [DFW97b], Ptolemy [Pto14], GEMOC [GEM19b], ou autres. Dans notre exemple, les composants de simulation sont des FMU, conformément au choix technologique de l'équipe projet MOISE. Il est à noter que les algorithmes, spécifiques à la technologie FMI, « `Global Master` » et « `Local Master` » entrent dans la catégorie « `SimulationService` » du métamodèle de la plateforme de simulation (figures 3.14 ou 3.19). Les squelettes et les paramètres de ces algorithmes de cadencement peuvent être automatiquement générés, sur la base de cette répartition. Le projet MOISE a expérimenté la génération de la configuration du bus de cosimulation « Cosimate » de l'entreprise CHIASTECK, partenaire du projet.

Lorsque les résultats des différentes simulations fonctionnelles sont conformes aux attentes de l'architecte système du produit, celui-ci a la possibilité de préciser ses intentions quant aux choix technologiques qu'il souhaite donner au mobile d'inspection. Ces choix apparaissent au niveau de la couche physique, ici, au niveau de la couche de construction (figure 3.20).

4.3.4 L'étape « Construction »

Dans l'étape fonctionnelle, l'architecte système du produit centre son attention sur les fonctionnalités. Dans cette étape physique (figure 3.20, couche « Const »), il étudie les différents choix technologiques possibles. Dans la couche fonctionnelle, le mobile d'inspection pouvait être multiforme : nacelle élévatrice, drone ou encore piéton. Ici, pour se conformer au choix de l'équipe projet MOISE, le mobile d'inspection est un drone à 4 hélices, dont le cas d'usage industriel est décrit à la section 4.2.

Pour l'aider dans sa tâche, l'architecte système du produit (SyA) dispose de l'ensemble des scénarios de test, issus de l'analyse de l'étape fonctionnelle (réutilisation « Amont »). À ces scénarios existants, il peut en ajouter d'autres, spécifiques à cette étape « Construction », pour conforter ses choix technologiques (objectifs de simulation numéro 01 & 02).

4.3.4.1 Produit : Étape construction – Espace de simulation : Exigences

À l'instar des exigences d'entrée pour l'étape fonctionnelle du produit, l'architecte de la simulation reçoit la demande de simulation, avec les exigences associées, en provenance de l'architecte système du produit. Comme pour l'étape fonctionnelle, l'objectif de la simulation est de vérifier que le choix technologique à 4 moteurs permet d'exécuter le plan de vol automatique sans encombre. Cet objectif constitue l'objectif de simulation Numéro 01.

Le modèle, transmis comme exigence, est représenté par la figure 4.6. Le modèle est bien réalisé au niveau physique. Il est possible d'y distinguer les différents éléments physiques tels que les cartes « `PC_Processor Inspection` », « `PC_Processor Motors` », etc.

Pour bien mettre en exergue le principe de la réutilisation « Amont », les fonctions de couleur verte sont les fonctions qui n'ont pas été modifiées, lors du passage du niveau fonctionnel au niveau physique. Dit autrement, ce sont des fonctions qui vont pouvoir être réutilisées lors de la simulation, puisque les modèles à base de code ou exécutables sont déjà présents (section 3.2.3.6, figure 3.8). À l'inverse, les fonctions de couleur orange sont les fonctions modifiées et adaptées aux choix technologiques de l'architecte physique du produit.

La fonction, en orange, « `Compute Thrust & Angles Consigns` » traduit les consignes de vitesses en consignes d'assiette du drone : roulis (Roll), tangage (Pitch), lacet (Yaw), et en puissance de poussée. Ces informations sont envoyées à la fonction « `Compute Movement Consigns` » qui assigne des vitesses de rotation à destination de la régulation de chaque moteur. De cette manière, en jouant sur les différentes vitesses de rotation des moteurs, le drone a la capacité de se déplacer suivant la pente, la direction et la vitesse voulue. Au niveau du bloc « `PC_Behavior_Motor-0x` », la fonction « `Create Mechanic Power` » correspond à un moteur, tandis que la fonction « `Create Movement` » correspond à l'hélice, qui par friction avec l'air, crée la force de poussée nécessaire au mouvement du drone.

4.3.4.2 Produit : Étape construction – Espace de simulation : Étape fonctionnelle, objectif 01

L'architecte de la simulation prend en compte les exigences d'entrée, représentées par la figure 4.6, pour les adapter à l'étape fonctionnelle de la simulation (figure 4.7), suivant les mêmes modalités que celles décrites au paragraphe 4.3.3.2. À l'instar de la figure 4.3, cet effort adaptation consiste à extraire les aspects fonctionnels des exigences d'entrées, à prendre en considération les sémantiques des fonctions, les liens de traçabilité et l'objectif de la simulation, dans le but de rechercher les allègements de puissances de calcul nécessaires à la simulation : suppression des fonctions « `Receiver Signal Position` » et « `Measure`

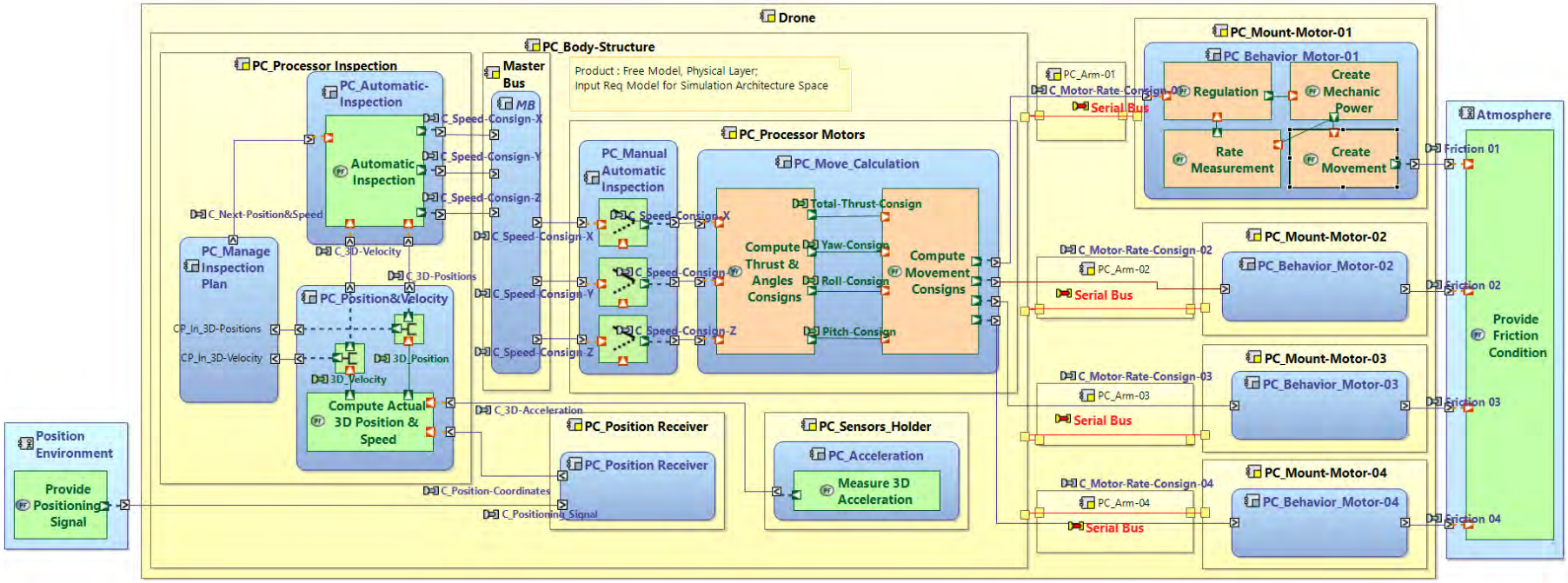

FIGURE 4.6 – Produit : Étape construction – Espace de simulation : Exigences d'entrée.

3D Acceleration ».

Pour ne pas altérer les résultats de la simulation, la fonction d'environnement « Provide Friction Condition » est modifiée pour tenir compte de la présence de ces quatre hélices muent par les quatre moteurs et pour fournir les informations nécessaires pour les fonctions « Compute Actual 3D Position & Speed » et « Provide Positioning Signal ». Les informations sont l'accélération représentée par le signal « 3D Acceleration » à destination de la fonction « Compute Actual 3D Position & Speed » et le déplacement du drone représenté par le signal « Movement Forces » pour la fonction « Provide Positioning Signal ». Ces signaux sont le reflet du déplacement du drone, tel qu'ils pourraient être mesurés par l'accéléromètre ou reçus du récepteur de positionnement, aux erreurs de simulation près.

4.3.4.3 Produit : Étape construction – Espace de simulation : Étape logique, objectif 01

À l'identique de la section 4.3.3.3, l'architecte de la simulation regroupe de manière logique, les fonctions entre elles, en accord avec les spécificités métiers de chaque partenaire. Les regroupements prennent en compte l'indépendance des 4 composants « Mechanical Move ». Le composant « LC_Move_Calculation » intègre les deux nouvelles fonctions. Quant aux autres composants, ils restent inchangés.

4.3.4.4 Produit : Étape construction – Espace de simulation : Étape physique, objectif 01

Cette étape physique pour la simulation (figure 4.9) est presque identique à l'étape physique de la simulation concernant l'étape fonctionnelle du produit (figure 4.5). Les différences se situent au niveau de certains partenaires qui accueillent les fonctions modifiées, ou en suppléments. Pour l'entreprise A, et sur l'unité d'exécution « Execution Unit 2 », la fonction initiale de la couche fonctionnelle est remplacée par les deux fonctions « Compute Thrust & Angles Consigns » et « Compute Movement Consigns ». Dans l'entreprise Z, la fonction de déplacement « Mechanical Move » est remplacé par les 4 FMU : « FMU Mechanical Move 01 » .. « FMU Mechanical Move 04 ».

Les composants de simulation FMU des unités exécution « Execution unit 1 » de l'entreprise « Enterprise A », et « Execution Unit 1 » de l'entreprise « Enterprise XY » restent inchangés. Cela signifie que les FMU développés, initialement pour la simulation du niveau fonctionnel, sont, ici, directement réutilisables. Il est à noter que la structure de la simulation, entre les partenaires, reste inchangée.

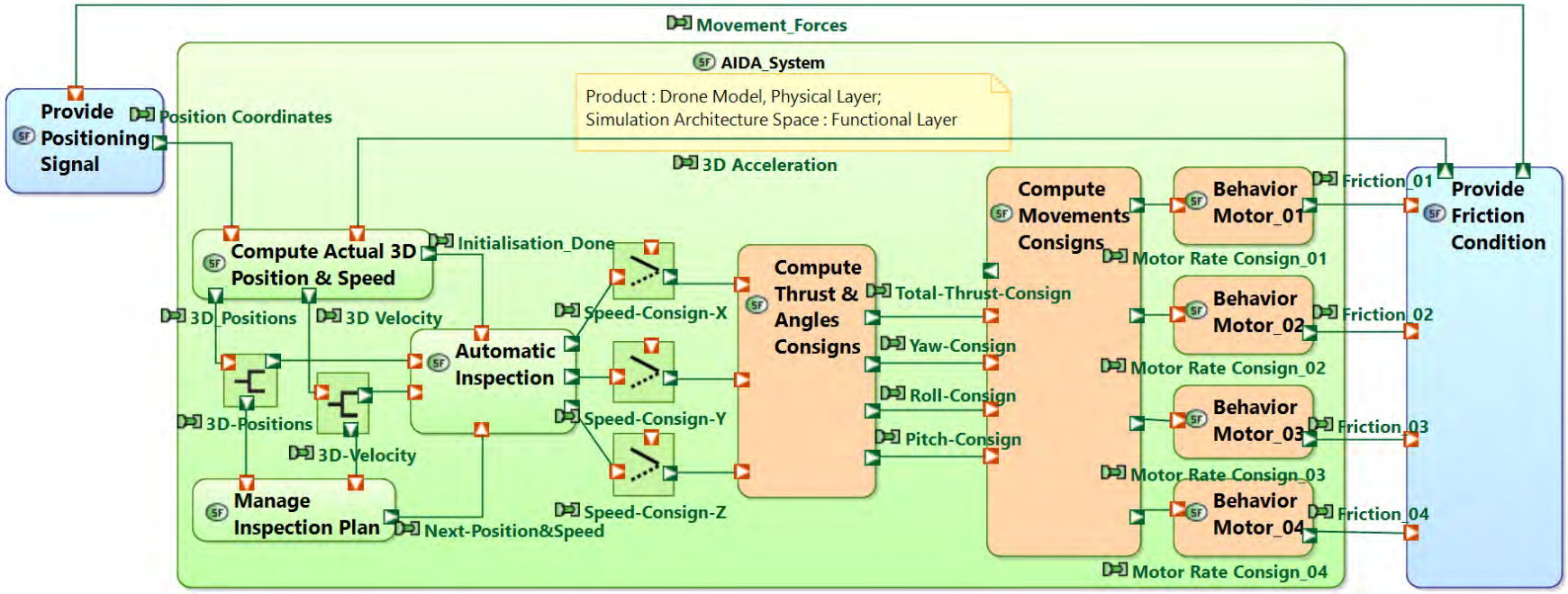

FIGURE 4.7 – Produit : Étape construction – Espace de simulation : Étape fonctionnelle, objectif 01

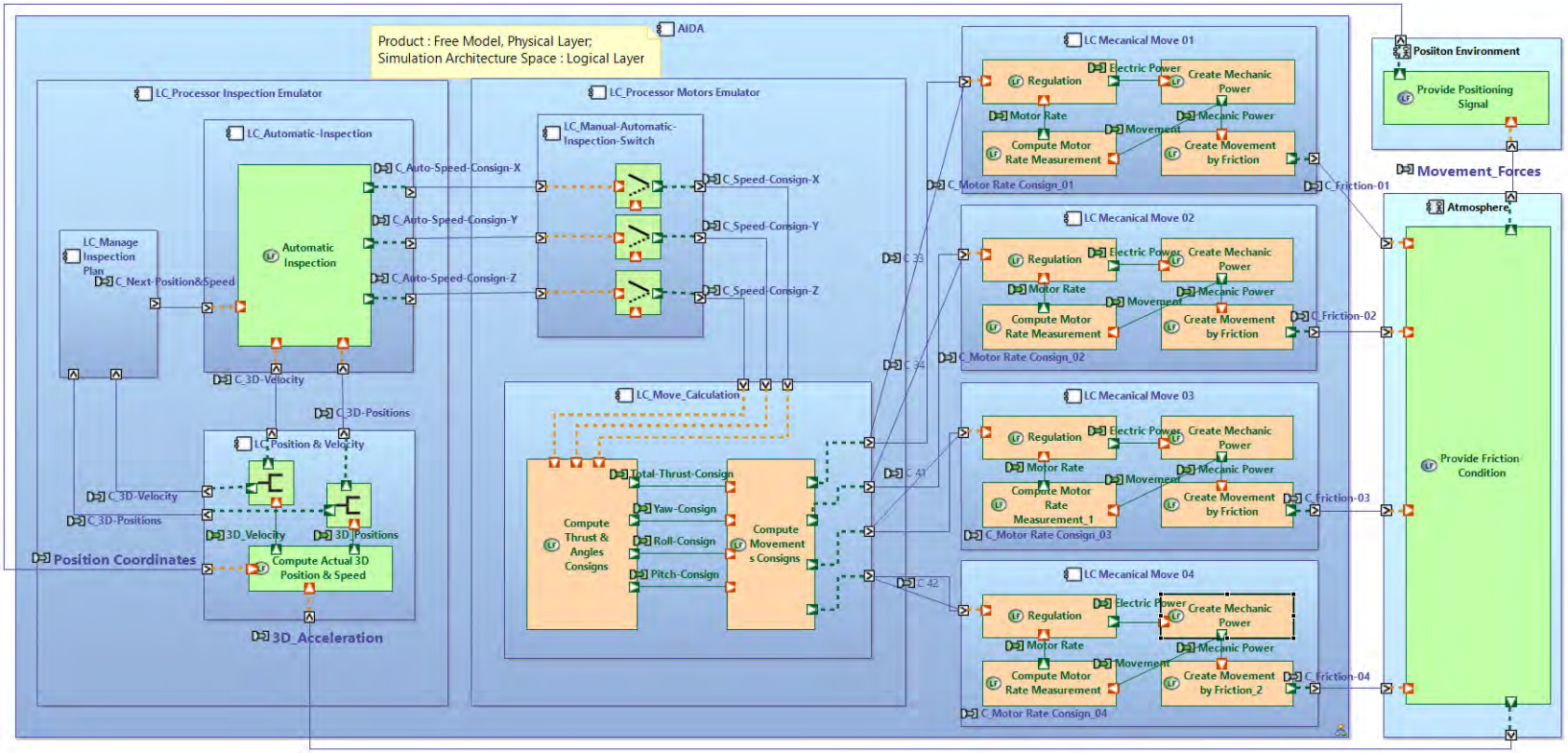

FIGURE 4.8 – Produit : Étape construction – Espace de simulation : Étape logique

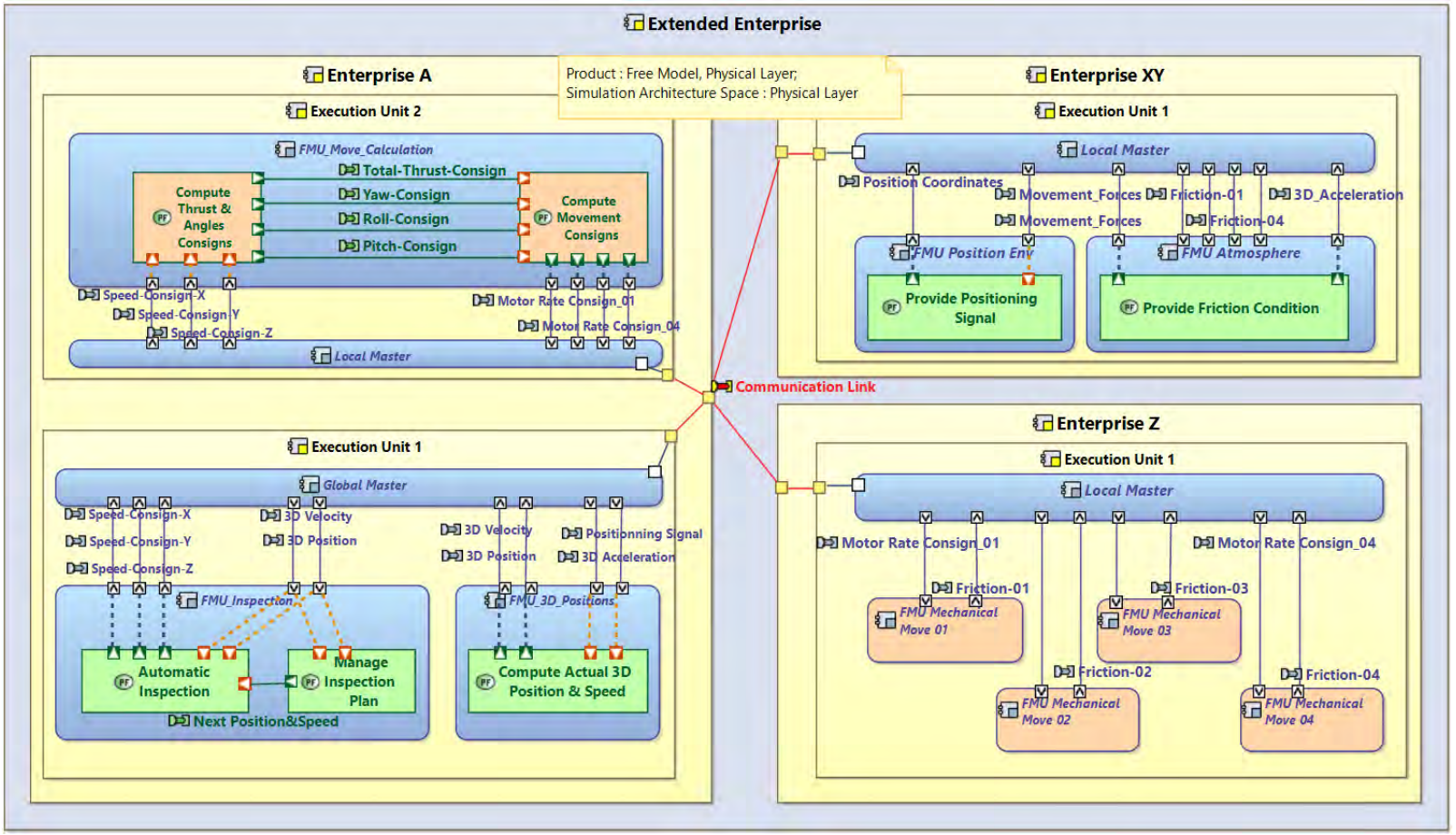

FIGURE 4.9 – Produit : Étape construction – Espace de simulation : Étape physique.

4.3.4.5 Produit : Étape construction – Espace de simulation : Exigences d'entrée, objectif 02

Pour illustrer les possibilités offertes par cette méthode, imaginons que l'architecte système pour le produit cherche à déterminer plus précisément les caractéristiques de l'architecture physique du drone. Parmi la diversité des tests envisageables, il est possible de citer les tests concernant la détermination approximative des temps d'exécution de chaque fonction, la marge de disponibilité de chaque processeur, la consommation électrique des processeurs faisant tourner les fonctions, la consommation électrique du drone, etc. Ici, pour l'architecte du produit, l'objectif Numéro 2 est de vérifier l'influence des délais introduits par les bus de communication « `Master Bus` » et « `Serial Bus` » (figure 4.6).

4.3.4.6 Produit : Étape construction – Espace de simulation : Étape fonctionnelle, objectif 02

Comme dans les étapes précédentes, l'architecte de la simulation reçoit l'objectif de la simulation, le modèle système et l'ensemble des exigences associées. Il traduit ces exigences sous la forme d'une architecture fonctionnelle dédiée à la simulation (figure 4.10). À l'identique de la section 4.3.4.2, on retrouve les fonctions, de couleur verte, inchangées par rapport au niveau fonctionnel, et les fonctions (couleur orange) modifiées ou ajoutées pour le choix technologique réalisé au niveau physique. À ces diverses fonctions s'ajoutent, ici, les fonctions (couleur rose) nécessaires pour répondre à l'objectif de simulation au niveau physique. La fonction « `Delay Bus` » simule le retard engendré par le bus de communication « `Master Bus` », tandis que les délais introduits par les bus de commande série des moteurs sont représentés par les 4 fonctions « `Delay Serial Bus` ». À charge pour l'architecte de la simulation de regrouper ces fonctions en des composants logiques.

4.3.4.7 Produit : Étape construction – Espace de simulation : Étape logique, objectif 02

Comme le travail sur cette étape logique l'exige, l'architecte de la simulation regroupe les fonctions entre elles, de manière logique. Ici, dans cet exemple, la fonction « `Delay Bus` », pour analyser l'effet du retard du bus « `Master` » sur la qualité de simulation, est allouée dans un unique composant logique « `Delay bus` » (figure 4.11). Le choix est fait pour les fonctions « `Delay Serial bus` » de les inclure dans les composants logiques dédiés à la motorisation.

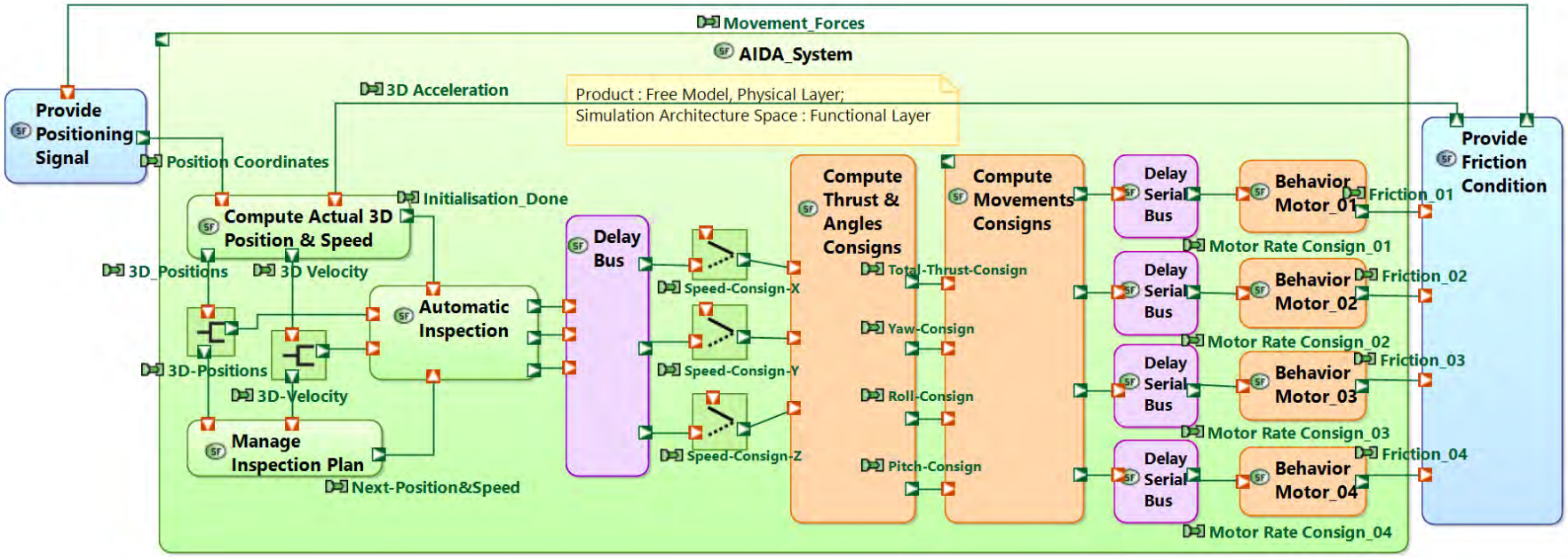

FIGURE 4.10 – Produit : Étape construction ; Espace de simulation : Étape fonctionnelle, objectif 02

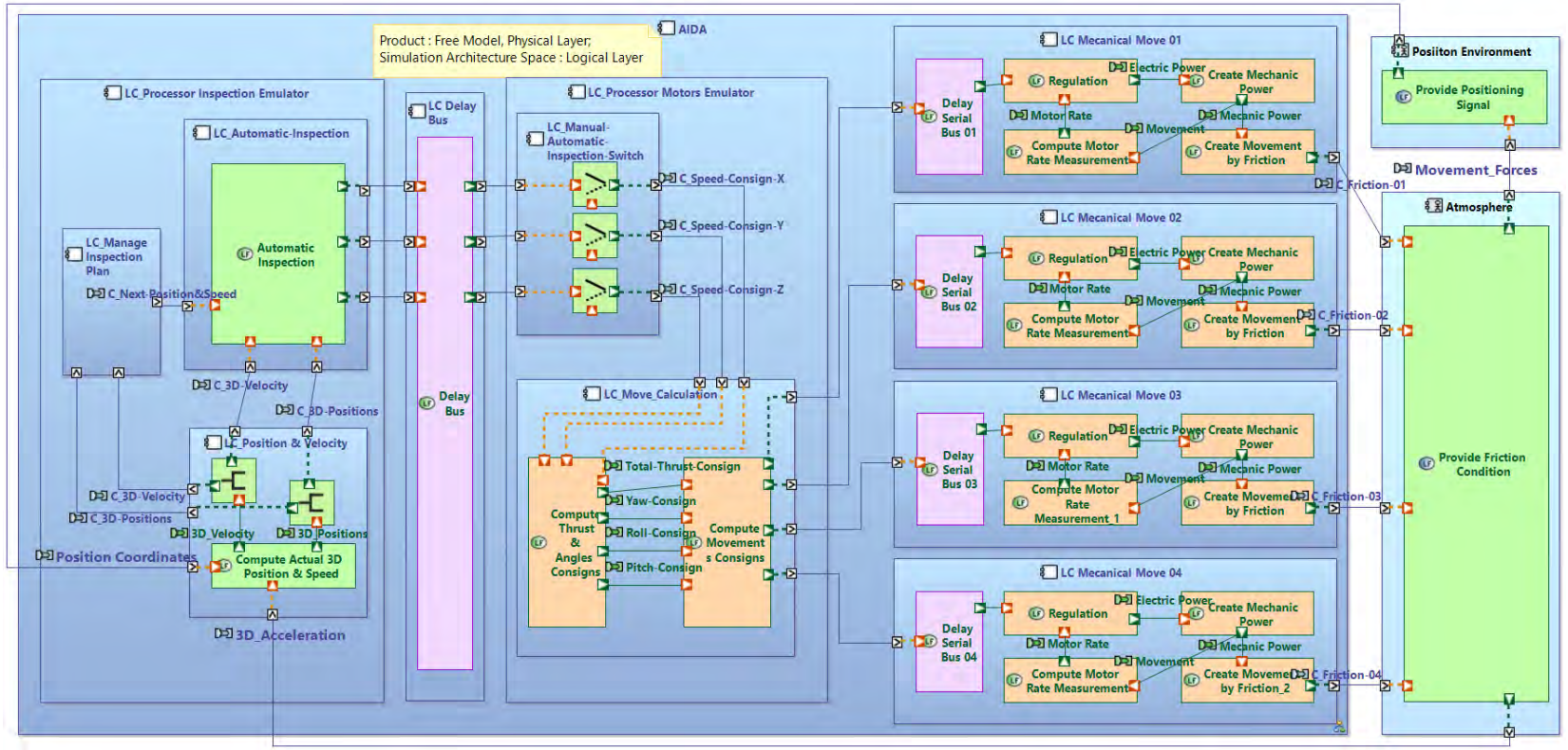

FIGURE 4.11 – Produit : Étape construction – Espace de simulation : Étape logique, objectif 02

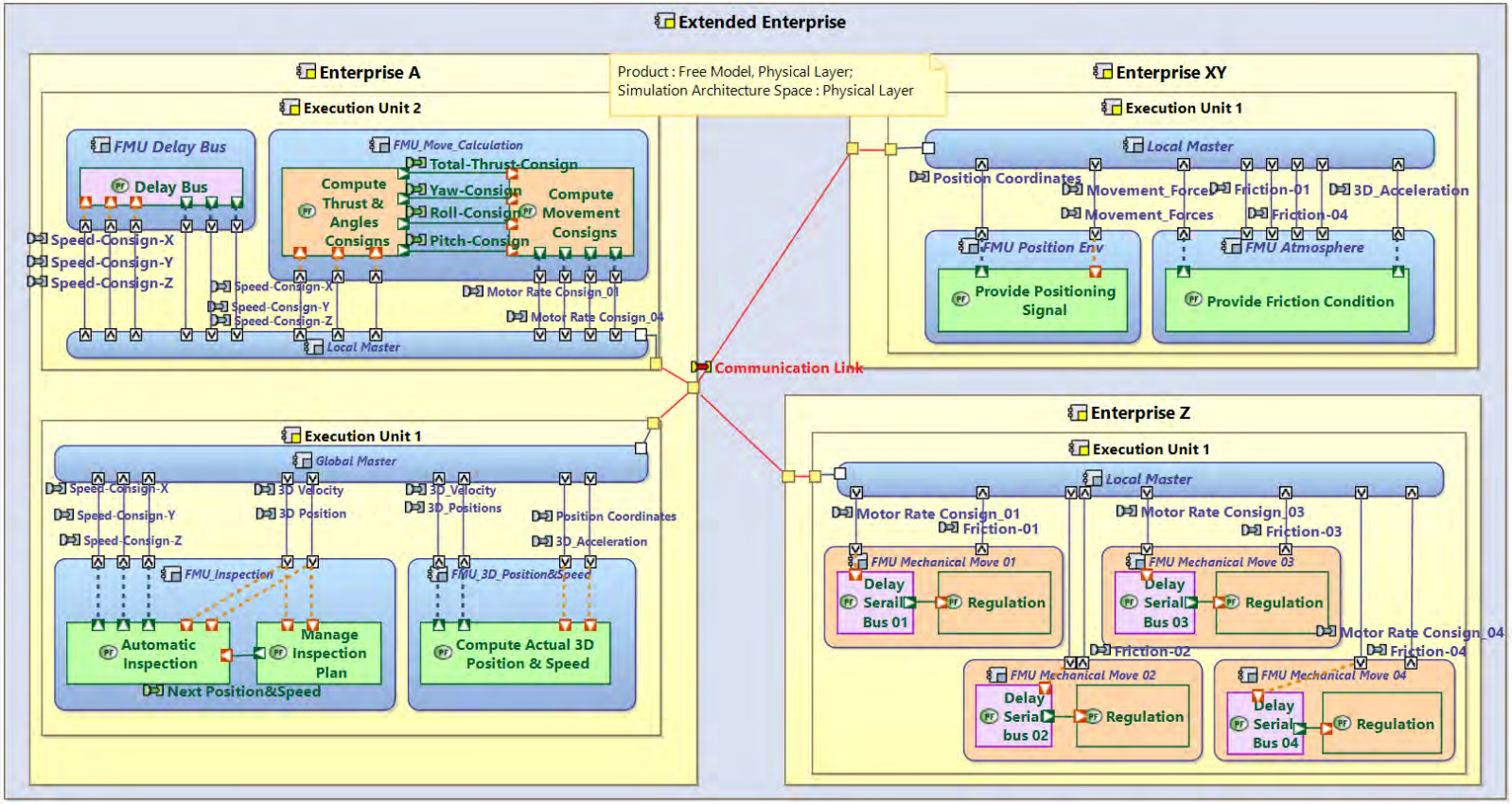

FIGURE 4.12 – Produit : Étape construction – Espace de simulation : Étape physique, objectif 02

4.3.4.8 Produit : Étape construction – Espace de simulation : Étape physique, objectif 02

Dans cette étape physique dédiée à la simulation, chaque composant logique est alloué à un équipement physique. Dans cette étape physique pour la simulation, les équipements sont de type FMU. La figure 4.12 représente les allocations sur ces équipements. Pour simplifier la lecture de cette figure, les FMU dédiés à la motorisation sont représentés avec leurs fonctions « `Delay Serial Bus 0x` » et la fonction « `Regulation` ». Bien évidement les autres fonctions y sont incluses sans y être représentées : « `Create Mechanic Power` », « `Compute Motor Rate Measurement` », « `Create Movement by Friction` ».

Sur cette base et en jouant sur la valeur des retards appliqués aux différentes fonctions « `Delay` », il devient possible d'évaluer leurs impacts sur la réponse du drone, quant à son positionnement, lors du vol d'inspection.

4.3.5 Exploitation à des fins prédictifs

La figure 4.7 présente des fonctions de couleurs différentes : verte et orange. Pour une couche d'analyse courante donnée, les fonctions de couleur orange sont les fonctions directement associées à cette couche d'analyse, tandis que les fonctions de couleur verte sont celles qui sont réutilisées, c'est-à-dire provenant d'une phase d'analyse « Amont ».

Or, cette même figure 4.7 peut être lue différemment, suivant une autre perspective. La couche d'analyse courante exploite les fonctions de couleur verte, tout en réutilisant les fonctions de couleur orange, issues d'une analyse « Aval » précédente, pour par exemple, démontrer plus finement la tenue d'un objectif de simulation. Bien évidemment, pour que la réutilisation « Aval » puisse être efficace (idem pour la réutilisation « Amont »), les API des fonctions doivent rester identiques entre les différents niveaux d'analyse.

4.4 Conclusion

Avant que l'architecte système du produit puisse gagner la confiance en la conception de son architecture, il y a des étapes nécessaires et indispensables qui doivent être parcourues.

En premier lieu, tout commence par l'assurance que la plateforme de cosimulation remplisse correctement son rôle (traitée à la section 3.5). A savoir la sécurité, la sûreté, les fonctionnalités matérielles, logicielles et les communications entre partenaires se doivent d'être normalement sans failles. A ces considérations générales, il est important d'ajouter

les vérifications & validations spécifiques à une simulation particulière : par exemple, le débit réseau, la disponibilité de la puissance de calcul, etc.

Une fois cette base établie, le déroulement de cette méthode sur le cas concret AIDA montre que cette méthode est applicable, à tout le moins pour le niveau système, et montre l'importance de la séparation des rôles entre l'architecte système du produit, l'architecte de la simulation, les développeurs et les personnes en charge de l'exécution de la simulation. De plus, il est montré que cette méthodologie permet de tenir la promesse de réutilisation des modèles entre les différents niveaux de raffinement au sein d'un même niveau système, ou entre niveaux système, à condition que les API des fonctions soient bien conservées.

Un autre point est l'importance de la vérification/validation du simulateur. Si le simulateur n'avait pas été considéré comme un projet en soi, avec ses différentes phases de tests unitaires, d'intégration, de vérification et validation, pour lever le doute quant à sa plénitude fonctionnelle, la moindre divergence entre les résultats attendus et ceux observés remettrait en cause dans un même élan englobant, tant la validité de ce simulateur, que la validité des scénarios de test, expression du besoin du client. Cette situation pourrait être potentiellement conflictuelle, tout en amenant des délais et coûts supplémentaires, pour vérifier et l'un et l'autre, avant de revoir l'analyse du produit. Cette approche considérant le simulateur comme un projet, tends à préserver les équipes, de cette situation embarrassante.

En fait, après exécution du cas de test AIDA, cette méthode tend à rationaliser le processus de développement des modèles d'un simulateur, en partant de l'expression des besoins de l'architecte système du produit, jusqu'à la simulation. Il serait souhaitable de pouvoir l'appliquer sur de plus amples cas concrets, pour pouvoir utiliser plus largement, voire généraliser, ce processus associé à la production de simulations, en partant des modèles du produit.

Chapitre 5

Conclusion et perspectives

5.1 Conclusion

Cette thèse propose une méthodologie dirigée par les modèles pour développer des outils de vérification et validation par la simulation en entreprise étendue, lors du développement d'un système en MBSE. Le but est d'aider l'architecte système du produit à définir son architecture avec confiance et dans le respect des exigences de besoins du produit. Cette méthodologie se décline suivant deux approches complémentaires l'une de l'autre : une méthode d'ingénierie système dirigée par les modèles destinée à la création de l'architecture de simulation, secondée par une modélisation de l'architecture physique de l'entreprise étendue, les réseaux de communication et les supports d'exécutions matériels et logiciels associés. Une illustration de haut niveau, de ces propositions, en est donnée par la figure 5.1, en se basant sur l'approche MDA. La branche PIM (Plateform Independant Model) contient la méthodologie qui a besoin de la branche PDM (Plateform Description Model) pour réaliser des simulations spécifiques (PSM : Platform Specific Model) à cette architecture de l'entreprise étendue.

Cette méthode pour réaliser des simulations peut être caractérisée par son approche matricielle, dans le sens où les colonnes représentent les acteurs intervenant dans la définition de l'architecture du produit (SyA : System Architect), de l'architecture de la simulation (SiA : Simulation Architect), du développement des modèles exécutables (SMD : Simulation Model developper) et de l'exécution de la simulation (SEM : Execution simulation Manager). Les lignes représentent les différentes étapes d'ingénierie système du produit, pour créer cette architecture système du produit. Contrairement à une approche classique où les exigences servent à définir l'architecture du produit et à spécifier les modèles de simulation, ici, le cheminement est différent : partir des exigences pour créer le modèle d'architecture système, puis à partir de ce modèle d'architecture, création de l'architecture de simulation, puis exécution la simulation en relation avec ses objectifs spécifiques. En agissant ainsi, les écarts d'interprétations potentiels entre le modèle d'architecture système et les modèles de simulation sont supprimés, ou à tout le moins réduits, par rapport à une approche traditionnelle.

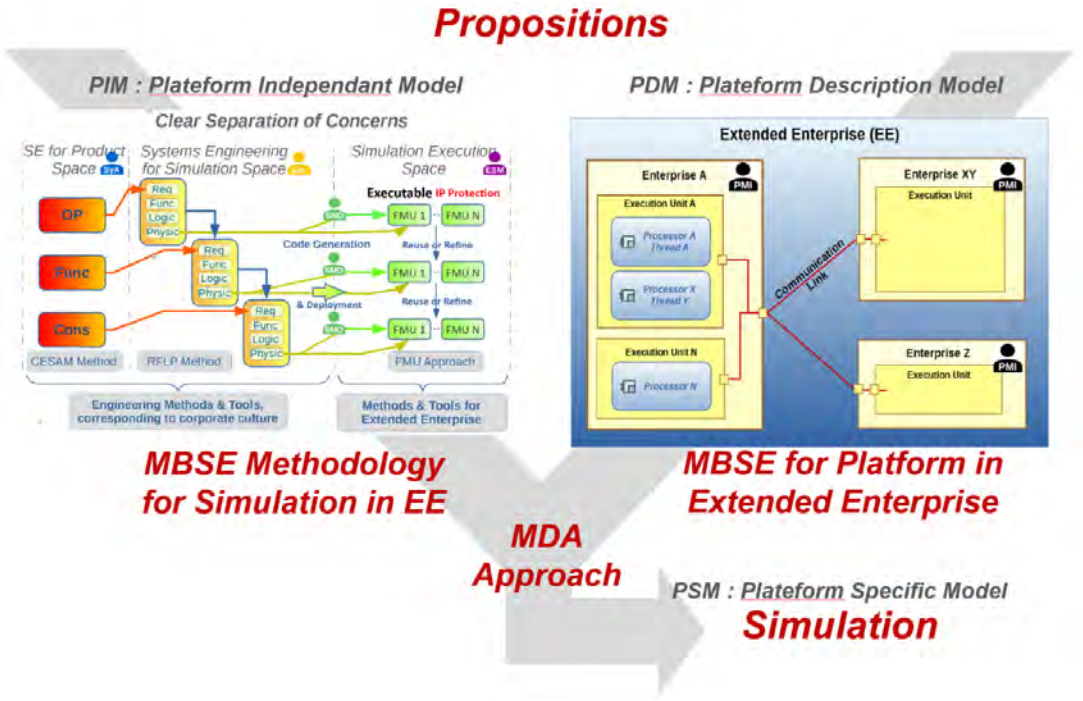

FIGURE 5.1 – Vue d'ensemble des propositions, avec l'approche MDA.

En complément de la méthode, des propositions d'exigences ont été formulées pour spécifier ce que peut être une plateforme de cosimulation en entreprise étendue. En effet, développer une telle plateforme est un projet en soi, tant pour le respect des coûts que des délais. À l'instar de la démarche d'ingénierie système qui se veut large d'approche, ces exigences couvrent la définition du rôle des acteurs, l'architecture de cette plateforme, la sécurité/sûreté, les matériels et les logiciels, et son intégration. Avec cette plateforme, il a été nécessaire de créer 3 nouveaux types d'acteurs. Les IPM « Infrastructure Project Manager », et dans le cadre de l'EE de type *fédéré*, les coordinateurs des IPM et des SEM. Cette méthodologie a été validée à l'aide du cas d'usage du projet MOISE : AIDA.

Les bénéfices de cette approche sont multiples. Cette méthodologie est compatible avec le principe de l'entreprise étendue. Elle offre une meilleure organisation du développement, par la mise en place de liens explicites entre les différents acteurs, particulièrement pour la traçabilité. Les acteurs sont indépendants les uns des autres, leurs rôles et responsabilités sont plus clairement définis. Cette méthode matricielle permet à l'architecte système du produit d'étudier son architecture tant en profondeur qu'en extension et de propager tout au long de la chaîne de conception des simulations, les liens de traçabilités. L'architecte de la simulation, quant à lui, a une grande souplesse pour construire son architecture de simulation. Cette souplesse est nativement inscrite dans l'aspect matriciel, et par les apports du groupe de travail du projet MOISE. Elle est également renforcée par la réutilisation amont ou aval de modèles de simulation exécutables déjà développés, lors d'explorations

précédentes. De plus, du fait de la modélisation de la plateforme de cosimulation, il a la possibilité d'assigner, en fonction de ses besoins, les modèles de simulation aux entreprises partenaires, en fonction de leurs spécificités. Enfin, un autre point important, c'est la préservation de la propriété intellectuelle des différents membres de cette entreprise étendue.

Toutefois, la méthode présente une difficulté. Les acteurs, notamment les architectes système pour le produit et pour la simulation sont indépendants. Ils peuvent utiliser l'outil de modélisation de leur choix ou celui spécifié par leur entreprise. Cela pose une difficulté. Elle concerne la transformation des modèles entre les outils des deux architectes. Même si cela est difficile sans être compliqué, c'est une situation qui peut rebuter. L'autre difficulté, c'est la nécessité pour l'architecte système pour le produit de fournir exactement les modèles pour la simulation nécessaire à la simulation et pour un objectif donné. Or, soit par la pression, soit par manque de connaissance, soit par indélicatesse, l'architecte système peut être amené à fournir l'ensemble de son modèle à l'architecte pour la simulation. À lui, de faire en sorte de construire son architecture de simulation en conservant les portions de modèles impliqués dans cette simulation. Le retour d'expérience de l'équipe MOISE pour la cosimulation permet maintenant de s'affranchir de ce fait, en fournissant les outils adéquats à l'architecte pour la simulation.

5.2 Perspectives

La méthodologie présentée dans cette thèse a été conçue pour le niveau « Système » dans le cycle en V. Une première perspective à ces travaux consisterait à réaliser des essais comparatifs entre une approche conventionnelle et l'approche proposée dans cette thèse, pour en déterminer plus objectivement les apports et les limites. Cette comparaison pourrait se faire sur un même sujet et dans des environnements industriels différents. La difficulté de cette perspective consiste en la possibilité de bénéficier de ressources matérielles, temporelles et financières pour une expérimentation réaliste. L'IRT Saint-Exupery capitalise les développements de tous les projets qu'elle réalise, notamment le projet MOISE auquel était adossée cette thèse. Dans le contexte de cet IRT, cette perspective de comparaison n'est pas irréaliste et pourrait intéresser des industriels et permettre de produire des articles scientifiques pour la communauté (méthodologie de sélection des critères de comparaison, évaluation, etc.).

Une perspective intéressante serait d'identifier les évolutions de cette méthodologie, basée MBSE et MDA, pour intégrer la notion de métriques et des mesures associées, pour caractériser les performances du système, lors des simulations. Ces métriques peuvent être de niveaux différents, par exemple : 1) au niveau du modèle système (identification et surveillance des paramètres clefs du système), 2) au niveau de la simulation (dérivé d'un

paramètre, excursion mini/maxi, temps de calcul, temps de latence sur le réseau de simulation, etc.). En première approche, il est possible de s'appuyer sur les travaux existant sur des métriques au niveau modèle logiciel [Mon+08], qui se basent sur une approche MDA pour générer automatiquement le code de mesure. Une autre approche structurée, complémentaire à la précédente, s'appuie sur une méthode de qualimétrie, qui consiste en l'identification des caractéristiques d'un système (aspect qualité) et en la quantification (aspect mesure) des paramètres identifiés [ABE19]. Cette méthode étend les standards ISO26262 :2011 (« Véhicules routiers - Sécurité fonctionnelle»), ARP4754A (Guide pour le développement d'aéronefs et systèmes civils) et DO-178C (Considérations sur le logiciel dans la certification des systèmes et équipements aéroportés) [Arg+20].

Une autre perspective possible serait d'étudier l'apport de cette méthode pour la simulation de « Système de Système » (SoS [1]). Dans ce cas, le rôle de l'architecte du produit évolue et devient l'architecte du SoS. C'est-à-dire qu'il va devoir définir l'architecture des interrelations entre les différents constituants du SoS, prendre en compte leurs propriétés respectives, décrire les comportements de chacun de ces éléments, tout en cherchant à préciser les comportements émergents souhaités [Oqu18]. Dans le cas de SoS pouvant évoluer dans le temps, il revient à l'architecte SoS de modéliser la configuration de l'architecture du SoS [PBB18], tout en anticipant sur les stratégies à employer pour prendre en compte les potentielles évolutions du SoS [PBB16]. A l'architecte de la simulation de mettre en œuvre ces évolutions. Dans le cas où le système de simulation (calculateurs et logiciels associés) est également considéré comme un SoS, l'architecte de la simulation devrait pouvoir définir les stratégies d'évolutions nécessaires, par exemple, à l'adjonction ou suppression de calculateurs. Il serait intéressant d'étudier les difficultés à surmonter lorsque les évolutions du SoS nécessitent l'ajout de calculateurs supplémentaires, en cours de simulation. Autre point à prendre en considération par l'architecte de la simulation, c'est la répartition des éléments du SoS sur cette architecture de simulation. Cette répartition ne devrait pas altérer le comportement émergeant du SoS, puisque d'après [Oqu18] le comportement émergeant peut être déduit (cas émergence faible) des comportements des éléments du SoS. Par exemple, pour une formation de drones plane ou en nuage, une architecture de simulation de type matricielle ou encore en 3D serait-elle bien appropriée ? Pour reprendre l'exemple de cette grappe de drones, l'architecte système peut-il découper cette grappe en sous-ensembles à appliquer à chaque nœud de simulation sans altérer le comportement émergeant, et à quelles conditions ? A cette dernière question, des réponses potentielles pourraient être trouvées par l'étude des stratégies de découpage mises en œuvre pour la simulation de l'écoulement des fluides (par exemple autour d'un réacteur) sur des cartes multiprocesseurs.

Un autre sujet d'étude serait d'appliquer cette méthodologie de conception de systèmes de simulations en entreprise étendue, sur la partie basse du cycle en V, au niveau équipe-

1. *System of System*

ment. L'idée serait de pouvoir réaliser des simulations temps réel. Dans ce contexte, quels seraient les éléments limitants dus à cette méthode et à l'infrastructure de simulation (par exemple : temps de calcul, temps de transfert des données, etc.) ? L'autre particularité serait de pouvoir intégrer, au fur et à mesure, les équipements réalisés dans cette architecture de simulation. Il serait également souhaitable de pouvoir mettre en place un mécanisme à base de modèles et méta-modèles pour pouvoir intégrer, dans cette méthodologie de simulation, des standards (par exemple : MARTE [2]) pour modéliser, définir et préciser les paramètres des éléments physiques à simuler : communication réseau, les accès mémoires, etc.

Dans la proposition de thèse, l'architecte du produit (SyA) utilise sa propre méthode d'ingénierie système et s'appuie sur un langage de modélisation pour exprimer sa pensée : par exemple SysML [3], ou encore et sans être limitatif, le langage fournit par le logiciel Capella. Or, les architectes métier sont parfois amenés à employer leurs propres langages pour traduire leurs préoccupations avec l'expressivité qui convient à leur domaine (par exemple, pour l'analyse de sûreté de fonctionnement MBSA [4]). Comme perspective, il semble vraiment intéressant de prendre en compte en entrée de cette méthodologie, différents langages de modélisation métier qui concourent à préciser les besoins des intervenants et à définir le système ou le système de système (SoS). Pour ce faire, l'OMG a fait appel à propositions (RFP [5]) pour définir des mécanismes d'extension de méta-modèles (MEF : Metamodel Extension Facility, non encore disponible pour les non-membres de ce groupe de travail). Dès publication, il serait instructif d'évaluer les impacts sur cette proposition méthodologique. Par ailleurs, [Sch15] propose dans le cadre de sa thèse le concept de *rôle* comme « *médiateurs dynamiques entre modèles système et modèles de simulation* ». Cette approche prend en compte la diversité des langages utilisés par les différentes entreprises partenaires d'un projet, pour décrire les systèmes constituants le SoS. De plus et comme avantage, l'approche de [Sch15] ne cherche pas à agréger, unifier ou encore fédérer les méta-modèles des langages des entreprises partenaires. Avec ce concept de *rôle* et de par leurs propriétés intrinsèques (par exemple, la composition), il semble possible d'obtenir la souplesse nécessaire à la sélection des éléments des modèles à destination de la simulation, pour par exemple, simuler des chaînes fonctionnelles. [Sch15] utilise dans sa thèse, le langage EFFBD [6] pour réaliser des simulations. Puisque les *rôles* sont des médiateurs, il semble également possible, pour une même demande de simulation, d'employer diverses technologies pour bénéficier des atouts de chacune d'elles : FMI, Gemoc, xCapella, et autres, sans devoir redéfinir l'ensemble des *rôles*. Le quatrième groupe du projet MOISE a créé un POC sous la forme d'une plateforme collaborative pour prendre en compte l'hétérogénéité des mé-

2. *Modeling and Analysis of Real-time and Embedded systems*
3. *Systems Modeling Language*
4. *Model-Based Safety Analysis*
5. *Request For Proposals*
6. *Enhanced Function Flow Block Diagram*

thodes, langages et outils de chaque partenaire dans l'entreprise étendue. Cette plateforme collaborative regroupe les modèles d'architecture système des différents partenaires, pouvant être des éléments d'entrée de cette proposition méthodologique. Adjoindre le concept de *rôle* médiateur permettrai d'avoir « aisément » une dimension supplémentaire qu'est la simulation, tout en ayant un couplage faible entre la plateforme collaborative (espace SyA) et la plateforme de cosimulation en entreprise étendue (espaces SiA, SMD et SEM).

Chapitre 6

Glossaire

Architecture [INC15]
- L'architecture définie une structure organisationnelle et les relations entre les différents éléments constitutifs d'un système.
- L'architecture peut être définie par : la structure organisationnelle d'un système ou d'un composant, ou encore par la structure organisationnelle d'un système et de ses directives de mise en œuvre (ISO/IEC 2009, 1) [INC15].

Complexe
Un système est un ensemble d'éléments en interaction. Il est considéré comme *complexe* lorsque la prédiction de son comportement ne peut se faire par l'analyse de ses constituants et de leurs interrelations, mais uniquement par l'expérience ou la simulation de la globalité du système.

Compliqué
Un système est un ensemble d'éléments en interaction. Il est dit *compliqué* lorsque l'analyse de chaque constituant et de leurs interactions permet de prédire son comportement.

Méthode [Roc07], [Est07]
Une méthode est attachée aux techniques de réalisation d'une tâche. Elle définit le « comment faire » de chaque tâche (dans ce contexte méthode, technique, pratique et procédure sont souvent interchangeables). Chaque méthode peut aussi aussi être, en elle–même, un processus avec sa séquence de tâches à réaliser selon des méthodes particulières. En d'autres mots, le « comment » d'un niveau d'abstraction devient le « quoi » du niveau inférieur.

Méthodologie [Roc07], [Est07]
Une méthodologie est définie comme une collection de processus, de méthodes et d'outils reliés. Une méthodologie est essentiellement une « recette » et peut être comprise comme l'application de processus, méthodes et outils relatifs à une classe de problèmes qui ont quelque chose en commun.

Modèle
Définition de Marvin L. Minsky : « *Pour un observateur* ***B****, un objet* ***A**** *est un modèle d'un objet* ***A*** *dans la mesure où* ***B*** *peut utiliser* ***A**** *pour répondre aux questions qui l'intéressent*

au sujet de ***A***. ».
Dit autrement, un modèle est une représentation, une abstraction de la réalité, suivant un point de vue particulier. Il doit pouvoir être questionné pour en évaluer sa justesse, ou pour répondre à des questions sur cette partie de la réalité modélisée, pour la comprendre, l'expliquer, ou en prédire son évolution.

Modèle de simulation
Un modèle de simulation est un modèle, réalisé sous la forme d'un programme informatique, destiné à simuler, plus ou moins fidèlement, un point particulier de la réalité.

Outil [Roc07], [Est07]
Un outil est attaché à une méthode particulière, pour augmenter l'efficacité de réalisation de la tâche. Le rôle d'un outil est de faciliter le « comment ».

Plateforme de simulation
Équipement matériel et logiciel destiné à recevoir et exécuter les modèles physiques ou logiciels de simulation.

Processus [Roc07], [Est07]
Un processus est une séquence logique de tâches réalisées pour atteindre un objectif particulier. Un processus définit ce qui est à faire sans préciser le « comment faire ». La structure d'un processus peut se présenter sous différents niveaux d'agrégation pour permettre les analyses et répondre aux différents besoins d'aide à la décision.

Sécurité
- Fait de ne pas être menacé par un danger. *Être en sécurité.* (Dictionnaire du logiciel « Antidote », correcteur orthographique et grammatical).
- Situation dans laquelle quelqu'un, quelque chose n'est exposé à aucun danger, à aucun risque, en particulier d'agression physique, d'accidents, de vol, de détérioration : Cette installation présente une sécurité totale [Lar18].

Simulateur
L'intégration des modèles exécutables de simulation sur la plateforme de simulation constitue le simulateur.

Sûreté
- Qualité d'une chose dont l'utilisation n'est pas susceptible de causer des blessures (Dictionnaire du logiciel « Antidote », correcteur orthographique et grammatical).
- Propriété de ce qui fonctionne de manière fiable et conforme à son type [CNR18].

Système
Un ensemble d'éléments intégrés, de sous-systèmes, ou d'assemblages qui accomplissent un objectif défini. Ces éléments incluent les produits (matériel, logiciel, microprogramme), des

processus, personnes, informations, techniques, infrastructures, services, et autres éléments de soutien [INC15].

Validation

La validation cible le développement du bon produit (« make the right product »). C'est à dire du produit qui correspond aux besoins de l'utilisateur - aux exigences implicites qui sont à la source de la rédaction de la spécification du produit [Boe84]. Cette validation peut être définit de manière informelle par la question : « Ai-je construit le bon produit ? » [Boe84].

Vérification

La vérification vise le développement du juste produit (« make the product right »). C'est-à-dire du produit qui satisfait à ses exigences explicites - la spécification du produit [Boe84]. De manière informelle, le terme « Vérification » pourrait être également défini via la question : « Ai-je bien construit le produit ? » [Boe84].

Bibliographie

[ABE19] Yann Argotti, Claude Baron et Philippe Esteban. « Quality quantification in Systems Engineering from the Qualimetry Eye ». In : *IEEE International Systems Conference (SysCon) (IEEE SysCon 2019)*. IEEE. Orlando, United States, avr. 2019 (cf. p. 146).

[AD03] Anas Abou El Kalam et Yves Deswarte. « OrBAC : un modèle de contrôle d'accès basé sur les organisations ». In : *Cahiers francophones de la recherche en sécurité de l'information* Numéro II, 1er trimestre 2003 (2003), pp30-43 (cf. p. 86).

[Alb+17] Alexandre Albore, Silvano Dal Zilio, Guillaume Infantes, Christel Seguin et Pierre Virelizier. « A Model-Checking Approach to Analyse Temporal Failure Propagation with AltaRica ». In : *Model-Based Safety and Assessment. IMBSA 2017*. T. 10437. Lecture Notes in Computer Science. Trento, Italy, sept. 2017, 15p. (Cf. p. 6).

[Alb+18] Alexandre Albore, Silvano Dal Zilio, Marie De Roquemaurel, Christel Seguin et Pierre Virelizier. « Timed Formal Model and Verification of Satellite FDIR in Early Design Phase ». In : *9th European Congress on Embedded Real Time Software and Systems (ERTS 2018)*. Toulouse, France, jan. 2018, 10p. (Cf. p. 6).

[ANR10] ANR. *Les Investissements d'avenir*. `https://anr.fr/fr/investissements-davenir/les-investissements-davenir/`. 2010 (cf. p. 5).

[Arg+20] Yann Argotti, Claude Baron, Philippe Esteban et Denis Chaton. « Quality Quantification Applied to Automotive Embedded Systems and Software Advances with qualimetry science ». In : *Embedded Real Time Systems (ERTS) 2020*. Toulouse, France, jan. 2020 (cf. p. 146).

[BDF18a] Jean-Paul Bodeveix, Arnaud Dieumegard et Mamoun Filali. « Event-B Formalization of a Variability-Aware Component Model Patterns Framework ». In : *15th International Conference on Formal Aspects of Component Software (FACS 2018)*. Pohang, South Korea, oct. 2018, p. 54-74 (cf. p. 6).

[BDF18b] Jean-Paul Bodeveix, Arnaud Dieumegard et Mamoun Filali. « Pattern-based requirements development (regular paper) ». anglais. In : *European Congress on Embedded Real-Time Software (ERTS 2018), Toulouse*. `https://www.erts2018.org/` : ERTS : Embedded Real Time Sofware et Systems, fév. 2018, (electronic medium) (cf. p. 6).

[Beu+15] Antoine Beugnard, Fabien Dagnat, Sylvain Guerin et Christophe Guychard. « Des situations de modélisation pour décrire un processus de modélisation ». In : *Revue des Sciences et Technologies de l'Information - Série ISI : Ingénierie des Systèmes d'Information* 20.2 (2015), p. 41-66 (cf. p. 34).

[BKC17] BKCASE. *Guide to the Systems Engineering Body of Knowledge (SEBoK).* Online, 07 janvier 2018. 2017 (cf. p. 41).

[Bla12] Bertrand Blancheron. *Maxi fiches de Sciences économiques, 2e éd. fiche 35 Le Taylorisme, p. 100-101.* Dunod, 2012 (cf. p. 14).

[Boe84] Barry W. Boehm. « Verifying and validating software requirements and design specifications ». In : *IEEE Software* (1984), p. 75-88 (cf. p. 151).

[Bos+18] Benjamin Bossa, Benjamin Boulbene, Sébastien Dubé et Marc Pantel. « Towards a Co-simulation Based Model Assessment Process for System Architecture ». In : *Software Technologies : Applications and Foundations*. Sous la dir. de Manuel Mazzara, Iulian Ober et Gwen Salaün. Cham : Springer International Publishing, 2018, p. 58-68 (cf. p. 7, 110, 112).

[Bou+05] Jean Bourrely, Patrice Carle, Michel Barat et François Lévy. *GENESIS : an integrated platform for designing and developing HLA applications.* `www.onera.fr/sites/default/files/Departements-scientifiques/DCPS/05E-SIW-013.pdf`. Jan. 2005 (cf. p. 47).

[Bou+11] Frédéric Boulanger, Cécile Hardebolle, Christophe Jacquet et Dominique Marcadet. « Semantic Adaptation for Models of Computations ». In : *Proceedings of the 11th International Conference on Application of Concurrency to System Design*. Sous la dir. de Benoît Caillaud, Josep Carmona et Kunihiko Hiraishi. IEEE Computer Society, juin 2011, p. 153-162 (cf. p. 55, 56).

[Bou+16] Erwan Bousse, Thomas Degueule, Didier Vojtisek, Tanja Mayerhofer, Julien Deantoni et Benoit Combemale. « Execution Framework of the GEMOC Studio (Tool Demo) ». In : *Proceedings of the 2016 ACM SIGPLAN International Conference on Software Language Engineering*. SLE 2016. Amsterdam, Netherlands, oct. 2016, p. 8 (cf. p. 57).

[CAP19] CAPELLA. *Open source MBSE tool to create system, software or hardware architectures.* `https://www.polarsys.org/capella/`. Online, 14 mai 2019. 2019 (cf. p. 6, 39, 110, 112, 120).

[CES19] CESAM. *Méthode CESAM.* `http://cesam.community/la-certification-cesam/`. Online, 05 juillet 2019. 2019 (cf. p. 6, 37, 76).

[Cha12] Thierry Charles. *Sous-traitance industrielle : quand l'histoire bégaie.* http://archives.lesechos.fr/archives/cercle/2012/03/28/cercle_45088.htm.
Online, 07 janvier 2018. 2012 (cf. p. 17).

[CHI18] CHIASTEK. *CoSimate : Cosimulation Operating Platform.* https://site.cosimate.com/. 2018 (cf. p. 110).

[Cla19] Clarity-se.org. *Arcadia, Define, analyse, Design & Validate System, software, Hardware Architectures.* http://polarsys.org/capella/arcadia.html. Juill. 2019 (cf. p. 38).

[CNR18] CNRTL. *CNRTL : Centre National de Ressources Textuelles et Lexicales.* Online, 25 juillet 2018. 2018 (cf. p. 72, 150).

[Com08] Benoît Combemale. *Ingénierie Dirigée par les Modèles (IDM), État de l'art.* https://hal.archives-ouvertes.fr/hal-00371565. Août 2008 (cf. p. 28-30, 33).

[Cré+13] Xavier Crégut, Marc Pantel, Benoit Combemale et Arnaud Dieumegard. *Ingénierie Dirigée par les Modèles, Méta-modélisation avec ECore et Transformation de Modèles en ATL.* http://cregut.perso.enseeiht.fr/ENS/2018-2sn-gls/CONTENU/gls-2sn-2018-cm-idm-sujet.pdf. 2013 (cf. p. 31, 32).

[DAC17] DACCOSIM. *DACCOSIM : Distributed Architecture for Controlled CO-SIMulation.* 2017 (cf. p. 51).

[Dea+14] Julien Deantoni, Papa Issa Diallo, Joël Champeau, Benoit Combemale et Ciprian Teodorov. *Operational Semantics of the Model of Concurrency and Communication Language.* Research Report RR-8584. INRIA, sept. 2014, p. 23 (cf. p. 58).

[DFW97a] Judith S. Dahmann, Richard M. Fujimoto et Richard M. Weatherly. « The Department of Defense High Level Architecture ». In : *Winter Simulation Conference.* 1997 (cf. p. 44).

[DFW97b] Judith S. Dahmann, Richard M. Fujimoto et Richard M. Weatherly. *The Department Of Defense **H**igh **L**evel **A**rchitecture (HLA).* 1997 (cf. p. 129).

[DFW98] Judith S. Dahmann, Richard M. Fujimoto et Richard M. Weatherly. « THE DoD HIGH LEVEL ARCHITECTURE : AN UPDATE ». In : *Winter Simulation Conference.* 1998 (cf. p. 44).

[DoD10] DoD. *Technology Readiness Levels in the Department of Defense (DoD).* https://www.army.mil/e2/c/downloads/404585.pdf. Online, 02 juin 2019. 2010 (cf. p. 5).

[Don04] Gérard Donnadieu. *SYSTEMIQUE et SCIENCE des SYSTEMES, Quelques repères historiques*. http://afscet.asso.fr/ (Rubrique : Textes de référence). Online, 05 janvier 2018. Mars 2004 (cf. p. 22).

[DSC12] DS/CATOD. *Guide HLA (High Level Architecture*. http://certificationhla-france.com/Doc_Certif/GuideHLA_DGA_Ed5.pdf\. 2012 (cf. p. 42, 46, 47).

[EIA99] EIA. *EIA-632 Standard*. http://www.psconsultech.com/yahoo_site_admin/assets/docs/EIA632.9212432.pdf. 1999 (cf. p. 22-26).

[ESI19] ESI. *SimulationX*. https://www.esi-group.com/software-solutions/system-modeling/simulationx. 2019 (cf. p. 110).

[Est07] Jeff A. Estefan. *Survey of model-based systems engineering (MBSE) methodologies*. INCOSE MBSE Focus Group, http://www.omgsysml.org/MBSE_Methodology_Survey_RevA.pdf. Mai 2007 (cf. p. 149, 150).

[Exe15] ExeceisConseil. *L'évolution des relations entre les donneurs d'ordres et les sous-traitants dans l'aéronautique (Partie 1)*. https://exeisconseil.com/supply-chain/evolution-des-relations-entre-les-donneurs-ordres-et-les-sous-traitants-dans. Jan. 2015 (cf. p. 16, 17).

[FIM15] FIM. *Guide pratique de l'usine du futur, enjeux et panorama de solutions*. http://industriedufutur.fim.net/wp-content/uploads/2015/11/Guide_pratique_UDF.pdf\https://ifm40.fr/lindustrie-4-0-cest-quoi/. 2015 (cf. p. 15).

[Fio12] Serge Fiorese. *Découvrir et comprendre l'Ingénierie Système*. AFIS - Cepaduès, mars 2012 (cf. p. 23, 24, 36, 41).

[FMI14] FMI. *Functional Mock-up Interface for Model Exchange and Co-Simulation, specification*. https://fmi-standard.org/downloads/. 2014 (cf. p. 48-50, 113, 114).

[FMI19] FMI. *Functional Mockup Interface*. http://fmi-standard.org/. 2019 (cf. p. 48, 129).

[Fon99] Fontanili. *Intégration d'outils de simulation et d'optimisation pour le pilotage d'une ligne d'assemblage multiproduit à transfert asynchrone*. http://perso.mines-albi.fr/~fontanil/THESE/. Theses. Sept. 1999 (cf. p. 13, 14).

[Fuj98] Richard M. Fujimoto. « Time Management in the High Level Architecture ». In : vol. 71 (1998), p. 388-400 (cf. p. 43).

[Gal+15] Virginie Galtier, Stephane Vialle, Cherifa Dad, Jean-Philippe Tavella, Jean-Philippe Lam-Yee-Mui et Gilles Plessis. « FMI-based Distributed Multi-simulation with DACCOSIM ». In : *Proceedings of the Symposium on Theory of Modeling and Simulation : DEVS Integrative Symposium*. DEVS '15. Alexandria, Virginia : Society for Computer Simulation International, 2015, p. 39-46 (cf. p. 49).

[Gar13] Jacques Garnier. « Evolutions de la sous-traitance industrielle et risques au travail ». In : *Chroniques du travail ISSN 2257-5650* Qualité du Travail, Emplois de Qualité (2013). HAL Id : halshs-01418057, `https://irt.univ-amu.fr/revue`, pp17-30 (cf. p. 16).

[GEM19a] GEMOC. *GEMOC Studio*. `https://projects.eclipse.org/projects/modeling.gemoc` . Sept. 2019 (cf. p. 58).

[GEM19b] GEMOC. *The GEMOC Initiative On the Globalization of Modeling Languages*. `http://gemoc.org/`. Online, 15 mai 2019. 2019 (cf. p. 129).

[Gol+16] Fahad Rafique Golra, Antoine Beugnard, Fabien Dagnat, Sylvain Guerin et Christophe Guychard. « Addressing Modularity for Heterogeneous Multi-model Systems using Model Federation ». In : *MODULARITY 2016 : 15th International Conference on Modularity*. Malaga, Spain, 2016, p. 206-211 (cf. p. 33).

[Gra+13] Pascal Graignic, Thomas Vosgien, Marija Jankovic, Vincent Tuloup, Jennifer Berquet et Nadège Troussier. « Complex System Simulation : Proposition of a MBSE Framework for Design-Analysis Integration ». In : *Procedia Computer Science* 16 (déc. 2013), p. 59-68 (cf. p. 41).

[Gue19] Janine Guespin-Michel. *La révolution du complexe*. `http://www.revolutionducomplexe.fr/emergence/sciences/40-structures`. 2019 (cf. p. 2).

[Hen+04] Sophie Henon, Sylvia Portut, Gérard Weisbuch et André Orléan. *Action concertée systèmes complexes en SHS (Sciences de l'Homme et de la Société)*. `http://www.cnrs.fr/prg/PIR/programmes-termines/systcomplex/appel2004.pdf`. 2004 (cf. p. 2).

[IEE05] IEEE. *1220-2005 - IEEE Standard for Application and Management of the Systems Engineering Process*. `https://ieeexplore.ieee.org/document/1511885`. 2005 (cf. p. 22).

[IEE10a] IEEEa. *IEEE Std 1516 - 2010 - Framework and Rules*. Août 2010 (cf. p. 44-46).

[IEE10b] IEEEB. *IEEE Std 1516 - 2010 - Federate Interface Specification.* Août 2010 (cf. p. 44, 45).

[IEE10c] IEEEC. *IEEE Std 1516 - 2010 - Object Model Template Specification.* Août 2010 (cf. p. 44, 45).

[IEE15] IEEE. *15288-2015 - ISO/IEC/IEEE International Standard - Systems and software engineering – System life cycle processes.* https://ieeexplore.ieee.org/document/7106435. 2015 (cf. p. 22).

[IET15] IETF. *Deprecating Secure Sockets Layer Version 3.0, RFC 7568* . https://tools.ietf.org/html/rfc7568. 2015 (cf. p. 86).

[INC15] INCOSE. *Systems Engineering Handbook, a guide for system life cycle processes and activities, Fourth edition.* WILEY, 2015 (cf. p. 22, 41, 59, 149, 151).

[INS18a] Institut National de la Statistique et des Etudes Economiques INSEE. *Sous-traitance de capacité, définition.* https://www.insee.fr/fr/metadonnees/definition/c1005. Online, 05 janvier 2018. 2018 (cf. p. 15).

[INS18b] Institut National de la Statistique et des Etudes Economiques INSEE. *Sous-traitance de spécialité, définition.* https://www.insee.fr/fr/metadonnees/definition/c1629. Online, 05 janvier 2018. 2018 (cf. p. 15).

[IRT19a] IRT-MOISE. *AIDA : Aircraft Inspection by Drone Assistant* . https://sahara.irt-saintexupery.com/AIDA/AIDAArchitecture. Mars 2019 (cf. p. 120).

[IRT19b] IRT-Plateforme. *IRT Saint Exupery, Experimental Platforms* . http://www.irt-saintexupery.com/platforms/. Juin 2019 (cf. p. 5).

[Jea12] Sébastien Leroyer Jean-Christophe Sauniere. *Innovation collaborative et propriété intellectuelle : quelques bonnes pratiques.* INPI, 2012 (cf. p. 116, 117).

[JHB14] Christophe Jacquet, Cécile Hardebolle et Frédéric Boulanger. « ModHel'X, un outil expérimental pour la modélisation multi-paradigmes ». In : *CIEL 2014.* Actes de la 3ième Conférence en Ingénierie du Logiciel. Paris, France, juin 2014, p. 115-118 (cf. p. 56, 57).

[Joe09] Nicolas Mouchnino Joel Thomas Ravix. « L'évolution de l'industrie aéronautique : les incidences de la production modulaire ». In : *La Revue de l'Ires, 2009/3 (n 62), p. 135-157.* *https://www.cairn.info/revue-de-l-ires-2009-3-page-135.htm*. 2009 (cf. p. 16, 17).

[Jud97] Richard M. Weatherly Judith S. Dahmann Richard M. Fujimoto. « The department of Defence High Level Architecture ». In : *Proceedings of the 1997 Winter Simulation Conference*. Institute of Electrical et Electronics Engineers (IEEE), 1997 (cf. p. 46).

[Jud98] Richard M. Weatherly Judith S. Dahmann Richard M. Fujimoto. « The DoD High Level Architecture : An Update ». In : *Winter Simulation Conference Proceedings*. Institute of Electrical et Electronics Engineers (IEEE), 1998 (cf. p. 45).

[Kay03] Alan Kay. *Dr. Alan Kay on the Meaning of "Object-Oriented Programming"*. `http://userpage.fu-berlin.de/~ram/pub/pub_jf47ht81Ht/doc_kay_oop_en`. Juill. 2003 (cf. p. 28).

[Kay93] Alan Kay. *The Early History of Smalltalk*. `http://gagne.homedns.org/~gagne/contrib/EarlyHistoryST.html`. 1993 (cf. p. 28).

[KK13] Sven Kleiner et Christoph Kramer. « Model Based Design with Systems Engineering Based on RFLP Using V6 ». In : *Smart Product Engineering*. Sous la dir. de Michael Abramovici et Rainer Stark. Berlin, Heidelberg : Springer Berlin Heidelberg, 2013, p. 93-102 (cf. p. 35).

[Kou+13] Ali Koudri, Christophe Guychard, Sylvain Guerin, Fabien Dagnat, Antoine Beugnard et Joël Champeau. « De la nécessité de fédérer des modèles dans une chaîne d'outils ». In : *Génie logiciel* 105 (juin 2013), p. 18-23 (cf. p. 33, 34).

[Lar18] Larousse. *Dictionaire Larousse*. Online, 25 juillet 2018. Larousse, 2018 (cf. p. 72, 150).

[Lee16] Edward Alan Lee. *Fundamental Limits of Cyber-Physical Systems Modeling*. `https://doi.org/10.1145/2912149`. Nov. 2016 (cf. p. 72, 73).

[Ler+17] Renan Leroux, Ileana Ober, Marc Pantel et Jean-Michel Bruel. « Modeling Co-simulation : A First Experiment. » In : *Proceedings of MODELS-2017, Satellite Event : Workshops* (sept. 2017) (cf. p. 7, 50, 104).

[Ler+18a] Renan Leroux, Marc Pantel, Ileana Ober et Jean-Michel Bruel. « CPS simulation models categories in Extended Enterprise ». In : *Proceeding of GEMOC workshop at the ACM/IEEE MODELS conference* (2018) (cf. p. 7).

[Ler+18b] Renan Leroux-Beaudout, Marc Pantel, Ileana Ober et Jean-Michel Bruel. « Model-Based Systems Engineering for Systems Simulation ». In : *Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2018)*. T. 11246. Limassol, Cyprus, oct. 2018, p. 429-448 (cf. p. 7).

[Les19] Nancy Lesinski. *3D Transportation, Dassault System.* https://blogs.3ds.com/3dsmobility/systems-engineering-in-todays-\automotive-landscape/. Juill. 2019 (cf. p. 35, 36).

[LW96] Barbara Liskov et Jeannette Wing. « A Behavioral Notion of Subtyping ». In : *ACM Transactions on Programming Languages and Systems* 16 (nov. 1996) (cf. p. 29).

[May+13] Tanja Mayerhofer, Philip Langer, Manuel Wimmer et Gerti Kappel. « xMOF : Executable DSMLs Based on fUML ». In : *Proceedings of the 6th International Conference on Software Language Engineering (SLE'13)*. T. 8225. Lecture Notes in Computer Science. Springer, 2013, p. 56-75 (cf. p. 58).

[Mey+13] Bart Meyers, Joachim Denil, Frédéric Boulanger, Cécile Hardebolle, Christophe Jacquet et Hans Vangheluwe. « A DSL for Explicit Semantic Adaptation ». In : *CEUR Workshop Proceedings* 1112 (sept. 2013) (cf. p. 55).

[MGC19] Salvador Martinez, Sebastien Gerard et Jordi Cabot. *On the Need for Intellectual Property Protection in Model-Driven Co-Engineering Processes.* https://modeling-languages.com/wp-content/uploads/2019/04/EMMSADIP\Protection.pdf. EMMSAD Conference, Rome, June 2019. 2019 (cf. p. 17).

[Min65] Marvin Minsky. *Matter, Mind and Models.* http://web.media.mit.edu/~minsky/papers/MatterMindModels.html. 1965 (cf. p. 27).

[Mod18] Modelica. *Modelica.* 2018 (cf. p. 52).

[Mon+08] Martin Monperrus, Jean-Marc Jézéquel, Joël Champeau et Brigitte Hoeltzener. « Measuring Models ». In : *Model-Driven Software Development : Integrating Quality Assurance*. Sous la dir. de Jörg Rech et Christian Bunse. IDEA Group, 2008 (cf. p. 146).

[Nee+14] Himanshu Neema, Jesse Gohl, Zsolt Lattmann, Janos Sztipanovits, Gabor Karsai, Sandeep Neema, Ted Bapty, John Batteh, Hubertus Tummescheit et Chandraseka Sureshkumar. « Model-Based Integration Platform for FMI Co-Simulation and Heterogeneous Simulations of Cyber-Physical Systems ». In : *Proceedings of the 10th International Modelica Conference ; March 10-12 ; 2014 ; Lund ; Sweden*. 96. Linköping University Electronic Press ; Linköpings universitet, 2014, p. 235-245 (cf. p. 49, 51).

[Oka+16] Topçu Okan, Durak Umut, Oğuztüzün Halit et Yilmaz Levent. *Distributed simulation, A Model Driven Engineering Approach.* Springer, Cham, 2016 (cf. p. 44, 45).

[OMG14] OMG. *MDA Guide revision 2.0.* https://www.omg.org/mda/. Juin 2014 (cf. p. 28).

[OMG16] OMG. *Meta Object Facility (MOF) Core Specification*. `https://www.omg.org/spec/MOF/2.5.1/PDF`. Nov. 2016 (cf. p. 30).

[Ope19] OPENFLEXO. *Openflexo technical infrastructure and research project.* https ://openflexo.org/. Online, 28 decembre 2019. 2019 (cf. p. 34).

[Oqu18] Flavio OQUENDO. « On the Emergent Behavior Oxymoron of System-of-Systems Architecture Description ». In : *13th Annual Conference on System of Systems Engineering (SoSE). IEEE, Paris, France* (2018), p. 417-424 (cf. p. 146).

[PBB16] Franck PETITDEMANGE, Isabelle BORNE et Jeremy BUISSON. « Assisting the evolutionary development of SoS with reconfiguration patterns ». In : *Sustainable Architecture : Global Collaboration, Requirements, Analysis (SAGRA)*. Copenhagen, Denmark : ACM, nov. 2016, p. 9 (cf. p. 146).

[PBB18] Franck PETITDEMANGE, Isabelle BORNE et Jérémy BUISSON. « Modeling System of Systems configurations ». In : *2018 13th Annual Conference on System of Systems Engineering (SoSE)*. Paris, France : IEEE, juin 2018, p. 392-399 (cf. p. 146).

[Piè10] Ludovic PIÈTRE-CAMBACÉDÈS. « Des relations entre sûreté et sécurité ». Theses. Télécom ParisTech (Page 11), nov. 2010 (cf. p. 85, 86).

[Pro+17] Tatiana PROSVIRNOVA, Estelle SAEZ, Christel SEGUIN et Pierre VIRELIZIER. « Handling consistency between safety and system models ». In : *IMBSA 2017 (International Symposium on Model-Based and Assessment)*. Trento, Italy, sept. 2017, p. 19-34 (cf. p. 6).

[Pto14] Claudius PTOLEMAEUS, éd. *System Design, Modeling, and Simulation using Ptolemy II*. `http://ptolemy.org/books/Systems`. 2014 (cf. p. 52-54, 129).

[Rei17] Philippe REITZ. *Modelica Un langage pour modéliser et simuler des systèmes dynamiques hybrides. Des systèmes cyber-physiques...* Équipe MAREL, `http://www.lirmm.fr/~reitz/Modelica/`. Nov. 2017 (cf. p. 52).

[res12] RESEARCHGATE. *What is the difference between complex and complicated ?* `https://www.researchgate.net/post/What_is_the_difference_between_\complex_and_complicated`. 2012 (cf. p. 2).

[Ret+13] Fabien RETHO, Hichem SMAOUI, Jean-Claude VANNIER et Philippe DESSANTE. « A model-based method to support complex system design via systems interactions analysis ». In : *Proceedings of the Posters Workshop at CSD&M* (2013) (cf. p. 41).

[Ret+14] Fabien Retho, Hichem Smaoui, Jean-Claude Vannier et Philippe Dessante. « Model of Intention : A concept to support models building in a complex system design project ». In : *ERTS* (2014), p. 115-126 (cf. p. 41, 51).

[Roc01] Guy Rocher. *"La mondialisation : un phénomène pluriel" in Daniel Mercure (dir), Une société monde ? Les dynamiques sociales de la mondialisation, Presses de l'Université Laval, De Boeck, 2001*
. `https://papyrus.bib.umontreal.ca/xmlui/bitstream/handle/1866/89/0046.pdf`
. 2001 (cf. p. 15).

[Roc07] Samuel Rochet. « Formalization of System Engineering Processes : Proposal of a method to adapt generic processes to different application contexts ». Theses. Université Paul Sabatier - Toulouse III, nov. 2007 (cf. p. 149, 150).

[Roq18] Pascal Roques. *Modélisation architecturale des systèmes avec la méthode Arcadia, guide pratique de Capella*. ISTE Group, 2018 (cf. p. 38, 39).

[Ror05] Brigitte Rorive. « L'entreprise réseau revisitée, une tentative d'ordonnancement des nouvelles formes d'organisation. » In : *Revue : Créer et Comprendre, Mars 2005* 79 (2005), p. 63-75 (cf. p. 19-21, 61).

[SAE10] SAE. *"Guidelines for Development of Civil Aircraft and Systems ARP4754A*
. `https://www.sae.org/standards/content/arp4754a/`
. 2010 (cf. p. 6).

[SAE96] SAE. *"Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne systems and Equipment ARP4761*
. `https://www.sae.org/standards/content/arp4761/`
. 1996 (cf. p. 6).

[Sar11] R. G. Sargent. « Verification and Validation of Simulation Models. » In : *In Proc ; 2011 Winter Simulation Conference.* (2011) (cf. p. 108, 109, 122).

[Sch15] Jean-Philippe Schneider. « Roles : dynamic mediators between system models and simulation models ». Theses. Université de Bretagne occidentale - Brest, nov. 2015 (cf. p. 147).

[She+04] Mark S. Shephard, Mark W. Beall, Robert M. O'Bara et Bruce E. Webster. « Toward Simulation-based Design ». In : *Finite Elem. Anal. Des.* 40.12 (juill. 2004), p. 1575-1598 (cf. p. 41).

[Sim19] Simulink. *Simulink User's Guide : Algebraic Loop Concepts*
. `https://fr.mathworks.com/help/simulink/ug/algebraic-loops.html`.
2019 (cf. p. 113).

[Sir+15] Goknur Sirin, Christiaan Paredis, Bernard Yannou, Eric Coatanéa et Eric Landel. « A Model Identity Card to Support Simulation Model Development Process in a Collaborative Multidisciplinary Design Environment ». In : *IEEE Systems Journal* (jan. 2015) (cf. p. 41).

[Sir15] Göknur Sirin. « Ingénierie des systèmes basés sur les modèles (MBSE) appliquée au processus de conception de simulation complexe : vers une ontologie de la modélisation et la simulation pour favoriser l'échange des connaissances en entreprise étendue ». Theses. Ecole Centrale Paris, mars 2015 (cf. p. 42).

[Ste+08] Dave Steinberg, Frank Budinsky, Marcelo Paternostro et Ed Merks. *EMF : Eclipse Modeling Framework, 2nd Edition*. 2008 (cf. p. 114).

[Sup14] Centrale Supelec. *ModHel'X — A framework for heterogeneous modeling*. `https://wdi.supelec.fr/software/ModHelX/`. Avr. 2014 (cf. p. 55).

[TH13] J. Stephen Topper et Nathaniel C. Horner. « Model-Based Systems Engineering in Support of Complex Systems Development ». In : *Johns Hopkins APL Technical Digest Volume 32, Number 1*. 2013 (cf. p. 41).

[Tin+07] Bruno Tinel, Corinne Perraudin, Nadine Thevenot et Julie Valentin. « La sous-traitance comme moyen de subordination réelle de la force de travail ». In : *Actuel Marx* premier semestre `https://halshs.archives-ouvertes.fr/halshs-00266368`.41 (2007), p. 153-164 (cf. p. 15).

[TUW13] TUWien. *xMOF, eXecutable MOF, is a metamodeling language*. `https://modelexecution.org/moliz/xmof/`. 2013 (cf. p. 58).

[Var+15] Matias Ezequiel Vara Larsen, Julien Deantoni, Benoit Combemale et Frédéric Mallet. « A Behavioral Coordination Operator Language (BCOoL) ». In : *International Conference on Model Driven Engineering Languages and Systems (MODELS)*. Sous la dir. de Timothy Lethbridge, Jordi Cabot et Alexander Egyed. 18. to be published in the proceedings of the Models 2015 conference. Ottawa, Canada : ACM, sept. 2015, p. 462 (cf. p. 58).

[Voi18] Jean-Luc Voirin. *Conception architecturale des systèmes basée sur les modèles avec la méthode Arcadia*. ISTE Group, 2018 (cf. p. 37).

[Wik19a] Wikipedia. *"Astrolabe planisphérique"* . `https://fr.wikipedia.org/wiki/Astrolabe_planisphérique`. 2019 (cf. p. 71).

[Wik19b] Wikipedia. *"Claude Ptolémée"* . `https://fr.wikipedia.org/wiki/Claude_Ptolémée`. 2019 (cf. p. 18).

Résumé — Ce manuscrit présente une méthodologie pour la conception de systèmes de simulation de modèles en entreprise étendue, basée sur l'ingénierie système dirigée par les modèles. Le but est de permettre à l'architecte système d'explorer des solutions alternatives et de vérifier et/ou valider l'architecture du système en cours de conception, en regard des exigences et besoins des parties prenantes. Cette méthodologie se décline suivant deux axes complémentaires : la partie méthode et les moyens d'exécution, sans lesquels il ne peut y avoir de simulation. Cette nouvelle méthode se fonde sur le principe suivant : partir des exigences utilisateur pour créer les modèles d'architecture système, puis en dériver l'architecture de simulation, développer les modèles exécutables et exécuter la simulation en relation avec les objectifs de vérification et/ou validation. En agissant ainsi, les écarts d'interprétations potentiels entre le modèle d'architecture système et les modèles de simulation sont supprimés ou à tout le moins réduits, par rapport à une approche traditionnelle. Cette nouvelle méthode est de type matriciel. Les colonnes représentent les acteurs, tandis que les lignes correspondent aux différentes étapes de la méthode MBSE employée par l'architecte système pour le produit, y compris les étapes de raffinements. Les acteurs sont l'architecte système pour le produit (SyA), un premier nouvel acteur introduit par cette méthode : l'architecte système pour la simulation (SiA), les développeurs des modèles exécutables de simulation (SMD). Un second nouvel acteur est en charge de l'exécution de la simulation (SEM) au sein de chacune des entreprises, en charge de l'analyse et de la production des résultats exploitables par l'architecte système pour le produit. Avec cette méthode matricielle, le SyA peut demander des simulations, soit en profondeur pour préciser un point particulier de son modèle, soit en extension pour vérifier la bonne concordance des fonctions entre elles, tout en réutilisant des fonctions déjà définies durant les étapes amont ou aval de ses décompositions précédentes. Au global, gains de temps, de coûts, et de confiance. Le deuxième axe de cette méthodologie concerne la réalisation d'une plateforme de cosimulation en entreprise étendue (EE), qui est un projet en soi. Le MBSE a permis de définir une architecture fonctionnelle et physique de cette plateforme de cosimulation qui peut être amendée en fonction des besoins exprimés par l'architecte de la simulation. La proposition introduit un troisième nouvel acteur : le Infrastructure Project Manager (IPM) qui est en charge de la coordination pour la réalisation de la plateforme de cosimulation, au sein de son entreprise. Pour une EE de type donneur d'ordres à sous-traitants, introduction de deux nouveaux acteurs : le superviseur d'IPM et le responsable de l'exécution des simulations (SEM), dont leurs rôles respectifs sont de faire le lien avec leurs pendants chez les partenaires.

Mots clés : méthodologie, approche matricielle, vérification, validation, simulation, cosimulation, plateforme cosimulation, FMI, entreprise étendue.

Abstract — This manuscript presents a methodology for the design of "early" simulations in extended enterprise, based on model-driven system engineering. The goal is to allow the system architect to explore alternative solutions, and to verify and/or validate the system architecture being designed, in relation to the user requirements. This methodology is divided into two complementary axes : the method part (new) and the means of execution, without which there can be no simulation. This new method is based on the following principle : starting from the user requirements to create the system architecture model, then derive the simulation architecture, develop the executable models and run the simulation in relation to objectives of verification and/or validation. By doing this, potential differences in interpretations between the system architecture model and simulation models are removed or at least reduced compared to a traditional approach. This method is of matrix type. The columns represent the actors, while the lines correspond to the different steps of the MBSE method used by the system architect for the product, including the refinement steps. The actors are the system architect for the product (SyA), a first new actor introduced by this method : the system architect for the simulation (SiA), the developers of the simulation executable models (SMD), and the second new actor in charge of the execution of the simulation (SEM). The analysis of its qualities and the production of results exploitable by the system architect for the product. As the method relies on a matrix structure, the SyA can request simulations, either in depth to specify a particular point of its model, or more in extension to check the good agreement of the functions between them. With this new matrix approach, the system architect for the product can reuse functions already defined during the upstream or downstream stages of its previous decompositions. Overall, saving time, costs, and confidence. The second axis of this methodology is the realization of an extended enterprise cosimulation (EE) platform, which is a project in itself. Based on a proposal of requirements specifications, the MBSE has defined a functional and physical architecture. The architecture of this platform can be modified according to the simulation needs expressed by the architect of the simulation. This is one of his prerogatives. The proposal introduces a third new player : the Infrastructure Project Manager (IPM) which is in charge of coordinating for the realization of the cosimulation platform, within his company. For an EE of federated type, that is to say from contractor to subcontractor, introduction of two new actors : - the supervisor of IPM, whose role is to link IPMs to solve the administrative and interconnection problems, - the person responsible in charge of the execution simulations. He coordinates, with the SEM of each partner, the implementation of simulations, ensures launches, and returns the results to all partners.

Keywords : methodology, matrix approach, verification, validation, simulation, cosimulation, cosimulation platform, FMI, extended enterprise

Court résumé — La simulation est essentielle pour le développement d'un système critique. Les activités de validation et de vérification des différents types de modèles produits, lors du développement d'un système, nécessitent la construction de nombreuses simulations ciblant les différents points de vue et phases de développement. L'objectif de cette thèse est de proposer et de valider une méthode pour le développement de tels simulateurs à partir de modèles système et des exigences de validation et de vérification (en appliquant une approche MBSE). Elle intègre la contrainte, dans le contexte de l'Entreprise Etendue, de la protection de la propriété intellectuelle, de l'architecture matérielle et logicielle du moyen de simulation étendu. Une attention particulière a été accordée à la cohérence des différents simulateurs associés aux différents points de vue, à la réutilisation des éléments entre eux et à la gestion des simulateurs tout au long du développement et de la maintenance des systèmes.

Short Abstract — Simulation is essential for the development of critical system. The validation and verification activities of the different types of models produced during the development of a system require the construction of many simulations dedicated to the different points of view and phases of development. The purpose of this thesis is to propose and validate a method for the development of such simulators from system models (by applying an MBSE approach) and validation and verification requirements. It integrates the constraint, in the context of the Extended Enterprise, of the protection of intellectual property, the hardware and software architecture of the extended simulation means. Special attention has been paid to the consistency of the different simulators associated with the different points of view, the reuse of elements between each other and the management of simulators throughout the development and maintenance of systems.